UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE-QUITO

CARRERA: INGENIERÍA ELÉCTRICA

Tesis previa a la obtención del título de: INGENIERO ELÉCTRICO EN LA CARRERA DE INGENIERÍA ELÉCTRICA

TEMA

"OPTIMIZACIÓN DE FLUJO DE POTENCIA EN EL SISTEMA ELECTRICO ECUATORIANO CON PROGRAMACION NO LINEAL BAJO MATLAB"

AUTOR

FREDDY SIMÓN QUILLE PINTO

DIRECTOR

ING. XAVIER ESPINOZA

Quito, 28 de Febrero de 2015

DECLARATORIA DE AUTORIA

Yo, Freddy Simón Quille Pinto a la Universidad Politécnica Salesiana a la publicación total o parcial de este trabajo de grado y su reproducción sin fines de lucro.

Además declaro que los conceptos y análisis desarrollados y las conclusiones del presente trabajo son de exclusiva responsabilidad del autor.

Quito, 28 de Febrero de 2015

_____

Freddy Simón Quille Pinto
C.I. 1712508249

AUTOR

CERTIFICADO

Certifico que la presente tesis fue realizada en su totalidad por Freddy Simón Quille Pinto bajo mi supervisión.

_____

Ing. Xavier Espinoza
Director de Tesis

AGRADECIMIENTO

A todos los docentes que supieron inculcar los conocimientos y enseñanzas que en la vida profesional son de mucha ayuda.

DEDICATORIA

A mi esposa y mi hermosa familia por estar presente en los momentos más difíciles y minutos de flaqueza, que han sido una motivación para continuar avanzando y poder superar cada obstáculo presentado.

ÍNDICE DE CONTENIDOS **PAG.**

DECLARATORIA DE AUTORIA I
CERTIFICADO II
AGRADECIMIENTO III
DEDICATORIA IV
ÍNDICE DE CONTENIDOS V
INDICE DE FIGURAS VII
INDICE DE TABLAS IX
Resumen X
Abstract XI
INTRODUCCION XII
CAPÍTULO I 1
EL ESTUDIO DE FLUJOS DE POTENCIA 1
1.1 Introducción 1
1.2 Antecedentes 4
1.3 Ecuaciones de flujos de potencia 6
1.3.1 Solución de las ecuaciones de flujo de potencia por el método de Gauss Seidel 10
1.3.2 Solución de las ecuaciones de flujo de potencia por el método de Newton Raphson. 11
1.3.3 Solución aproximada de los flujos de potencia. 14
1.4 Conceptos Básicos 15
1.4.1 Potencia eléctrica, activa y reactiva. 15
1.4.2 Potencia Compleja. 21
1.4.3 Triángulo de potencias. 22
1.4.4 Sistema Por Unidad 23
1.4.5 El Diagrama Unifilar o de una Línea. 26
CAPITULO II 28
TÍTULO: FORMACIÓN DE LA MATRIZ DE ADMITANCIA YBUS 28
2.1 Introducción. 28
2.2 Estudio de la Admitancia. 29
2.3 Formación de la matriz de Admitancias, aplicación de las de Kirchhof 31
2.3.1 Formación de la Matriz de Admitancia por las aplicaciones de matrices de Transformaciones e inspección de la Red. 38
2.4 Singulares, Ejemplos 44

CAPITULO III 48
PROGRAMACIÓN NO LINEAL BASADO EN MATLAB 48
3.1 Programación no lineal 48
3.2 Consideraciones necesarias para la programación no lineal. 53
3.3 Estudio de Flujo de potencia óptimo. 54
3.4 Simulaciones con Matlab 56
3.4.1. Introducción 56
3.4.2 Comparación con Software actuales 58
3.5 PSAT (Power System Analysis Toolbox) 59
3.5.1 Gui Principal 59
3.5.2 Modelos Definidos por el Usuario 65
3.5.3 Modelos y Rutinas PSAT 71
CAPITULO IV 78
FLUJO ÓPTIMO DE POTENCIA 78
4.1 Formulación. 78
4.2 Casos de estudio 87
4.3 Análisis de resultados 89
CONCLUSIONES Y RECOMENDACIONES. 90
Referencias: 94

INDICE DE FIGURAS

Figura 1.1: Estructura de un Sistema de Potencia. 3
Figura 1.2: Presentación del menú principal y la base de datos. 4
Figura 1.3: Aplicación GDXVIEWER. 5
Figura 1.4: Exportación mediante tablas hacia EXCEL. 5
Figura 1.5: Aproximación xi a la raiz xy de f(x). 9
Figura 1.6: Modelo de línea de transmisión. 2
Figura 1.7: Variación con respecto al tiempo de v,i y p para la resistencia. 6
Figura 1.8: Variación con respecto al tiemp de v,i y p para una inductancia. 7
Figura 1.9: Variación con respecto al tiempo de v,i y p para una capacitancia. 8
Figura 1.10: Circuito electrico con elementos R,L y C. 19
Figura 1.11: Variación con respecto al tiempo de v,i y p para un circuito RLC..... 20
Figura 1.12: Circuito eléctrico monofásico. 21
Figura 1.13: Triángulo de potencia. 22
Figura 1.14: Circuito trifásico balanceado. 26
Figura 1.15: Diagrama unifilar de una Subestación Eléctrica 27
Figura 2.1: Vectores de susceptancia. 30
Figura 2.2: Equivalencia serie o paralelo. 30
Figura 2.3: Sistema de potencia compuesto por 3 buses. 32
Figura 2.4: Sistema representado por sus impedancias. 32
Figura 2.5: Sistema eléctrico representado por sus admitancias.. 33
Figura 2.6: Nodo 2 y Nodo 2 34
Figura 2.7: Desarrollo del nodo 1. 35
Figura 2.8: Admitancia en nodo 1. 35
Figura 2.9: Corrientes de fuentes. 37
Figura 2.10: Resultado de voltaje y corriente desarrollado en Matlab. 38
Figura 2.11: Determinación de la matriz YBus desarrollando en Matlab. 38
Figura 2.12: Resutado numérico aplicando Matlab. 38
Figura 2.13: Simulación de circuito aplicando Matlab. 39
Figura 2.14: Red eléctrica de tres nodos. 40
Figura 2.15: Nodo 1 se muestran corrientes que entran y salen..... 41
Figura 2.16: Nodo 2 se muestran corrientes que entran y salen 42
Figura 2.17: Nodo 3 se muestran corrientes que entran y salen 42
Figura 2.18: Circuito eléctrico ejercicio 1. 46
Figura 2.19: Resumiendo valores del ejercicio 1. 47

Figura 2.20: Circuito eléctrico ejercicio 2. 47
Figura 3.1: Ubicación inicio de un GUI 64
Figura 3.2: Ventana inicio de GUI. 64
Figura 3.3: Flujo de opción GUI. 66
Figura 3.4: Ventana de comandos Matlab. 66
Figura 3.5: Ventana de programación de GUI. 67
Figura 3.6: Ventana de comandos. 67
Figura 3.7: Interfaz de GUI. 68
Figura 3.8: Bloque de función definida por el usuario Fcn. 69
Figura 3.9: Bloque de una función definida por el usuario MATLAB Function……70
Figura 3.10: Bloque de función definida por el usuario S-Function. 71
Figura 3.11: Diagrama de la relación de entradas, estados y salidas en un modelo de Simulink. 72
Figura 3.12: Etapas de ejecución de una S-Function. 73
Figura 3.13: Bloque constructor de las S-Funciones, S-Function Builder. 74
Figura 3.14: Soporte para GUI. 76
Figura 3.15: Librería que presenta Simulink en Matlab. 77
Figura 3.16: Esquema sinóptico de PSAT. 80
Figura 4.1: Red eléctrica de 4 nodos. 82
Figura 4.2: Código de programación Matlab. 83
Figura 4.3: Código de programación Matlab. 85
Figura 4.4: Ventana principal para la programación. 87
Figura 4.5: Ventana principal para la programación. 88
Figura 4.6: Ventana editable cambiando propiedades. 89
Figura 4.7: Ventana editable donde se despliega la solución. 90
Figura 4.8: Ventana de iteracciones definida. 91
Figura 4.9: Ventana donde se verifica la solución óptima del sistema. 91
Figura 4.10: Potencia Reactiva con Flujo Óptimo de Potencia 92
Figura 4.11: Potencia Activa con Flujo Óptimo de Potencia. 92
Figura 4.12: Potencia Activa con Flujo Normal. 93
Figura 4.13: Potencia Reactiva con Flujo Normal. 93

INDICE DE TABLAS

TABLA 2. 1: Informacion de interconexión de la red. 43

TABLA 2.2: Paquetes basados en MATLAB para el análisis de sistemas eléctricos de potencia. 61

TABLA 2.3: Modelos típicos de simulación de PSAT. 76

TABLA 2.4: Comparación del soporte de Matlab y Octave con PSAT. 81

Resumen
“Optimización De Flujo De Potencia En El Sistema Eléctrico Ecuatoriano Con Programación No Lineal Bajo Matlab”

Freddy Simón Quille Pinto
freddquillfq@gmail.com
Universidad Politécnica Salesiana

Resumen—El tema propuesto para el desarrollo de este trabajo es determinar un punto óptimo de operación del Sistema Eléctrico Ecuatoriano, para analizar y minimizar las pérdidas, esto implica el estudio del flujo de potencia óptimo que tiene la ventaja de poder resolver despachos económicos y minimizar las pérdidas simultáneamente. Entre otros puntos importantes, e flujo de potencia es capaz de resolverlos, y para este trabajo se ha utiliza para la resolución, el método de Newton Raphson, este método requiere primeramente de una división de variables de control y variables de estado, luego se hace necesario definir ciertas funciones objetivo, para luego si aplicar el método con el objeto de obtener iterativamente, correcciones sucesivas sobre las variables de control, hasta llegar a un punto de operación del sistema de potencia, en el cual las funciones objetivo planteadas, tienen un valor óptimo.

El presente trabajo tiene como objetivo obtener un modelamiento de flujo óptimo de potencia y consecuentemente, un programa en MATLAB capaz de resolverlo. Para el análisis del tema se analizó los flujos de potencia: activa, reactiva.

Índice de Términos— Flujo Óptimo De Potencia, Voltaje, Método De Newton Rapshon, Corriente, Potencia Activa, Reactiva, Optimización De Flujo, Método Jacobiano.

Abstract
“Power Flow Optimization In The Ecuadorian Electric System With Nonlinear Programming Under Matlab”

Freddy Simón Quille Pinto
freddquillfq@gmail.com
Universidad Politécnica Salesiana

Abstract— The theme proposed for the development of this work is to determine optimal operating point of the Ecuadorian Electric System to analyze and minimize losses, this involves the study of optimal power flow has the advantage of solving economic offices and minimize losses simultaneously. Other important points, and power flow is able to solve, and this work has been used for the resolution, the method of Newton Raphson, this method requires first a division of control variables and state variables, then it becomes necessary to define certain objective functions, then if the method applied in order to obtain iteratively, successive corrections for the control variables, up to a point of operation of the power system, which raised objective functions have a value optimal.

This work aims to obtain optimum flow modeling power and consequently, a program in MATLAB able to solve. For analysis of the subject power flows active, reactive analyzed.

Index of Terms— Optimal Power Flow, Voltage, Method Newton Rapshon, Current, Active Power, Reactive Flow Optimization, Jacobian method.

INTRODUCCION

A partir del descubrimiento de la energía eléctrica y su posible utilización comercial por parte del hombre, ésta ha jugado un papel muy importante en el desarrollo de la humanidad. El desarrollo de grandes y eficientes fuentes de energía para ejecutar trabajos útiles ha sido la clave del dilatado progreso industrial y parte primordial en la mejora de la calidad de vida del hombre, en la sociedad moderna.

Pero el proceso de hacer llegar la energía eléctrica, desde las fuentes hasta los consumidores, requiere de estructuras cada vez más complejas, denominadas Sistemas de Potencias este trabajo presenta una manera óptima de modelar un algoritmo con el objetivo de optimizar y presentar una alternativa para minimizar las perdidas.

CAPÍTULO I

EL ESTUDIO DE FLUJOS DE POTENCIA

1.1 Introducción

El flujo de potencia es la denominación que se da a la solución de estado estacionario de un sistema de potencia bajo ciertas condiciones pre establecidas de generación carga y topología de la red [1].

A partir del descubrimiento de la energía eléctrica y su posible utilización comercial por parte del hombre, ésta ha jugado un papel muy importante en el desarrollo de la humanidad.

El desarrollo de grandes y eficientes fuentes de energía para ejecutar trabajos útiles ha sido la clave del dilatado progreso industrial y parte primordial en la mejora de la calidad de vida del hombre, en la sociedad moderna.

Pero el proceso de hacer llegar la energía eléctrica, desde las fuentes hasta los consumidores, requiere de estructuras cada vez más complejas, denominadas Sistemas de Potencias. Las cuales poseen asociadas una serie de fenómenos en condiciones operativas normales y anormales como por ejemplo:

- Fenómenos transitorios ultrarápidos: Corresponden sustancialmente a descargas atmosféricas sobre las líneas de transmisión y a los fenómenos producidos por operaciones de conexión y desconexión de diversos componentes de la red del SEP.

Las perturbaciones de este tipo dan origen a ondas de tensión y corriente que viajan prácticamente a la velocidad de la luz, pero su efecto dura unos pocos milisegundos después de iniciado. Sin embargo, los procesos de reflexión de las ondas producen elevadas tensiones que pueden llegar a destruir el equipo asociado a las líneas. La razón del estudio de estos fenómenos radica en el hecho de que su análisis suministra las bases necesarias para la selección adecuada del nivel de aislación de los equipos eléctricos asociados a las líneas y de las líneas mismas.

- Fenómenos transitorios medianamente rápidos: En este grupo se incluyen los fenómenos causados por cambios abruptos de la estructura del SEP, o sea los cortocircuitos o líneas abiertas. Usualmente, sólo los 10 primeros ciclos son de importancia práctica y se estudian en el rango de 10 a 100 milisegundos siguientes a la falla.

- Fenómenos transitorios lentos: Cuando ocurre un cortocircuito en una línea de transmisión importante y no se desconecta oportunamente la sección afectada, puede producirse uno de los fenómenos más peligrosos de un SEP, esto es, oscilaciones mecánicas de los rotores de los generadores. Se producen fenómenos transitorios electromecánicos que se estudian bajo el nombre de estabilidad transitoria. Las oscilaciones mecánicas de los rotores son relativamente lentas, en consecuencia, los estudios de estabilidad transitoria se realizan en el rango de fracción de segundo hasta un minuto.

Debido a los fenómenos transitorios pueden producir en un SEP, diversas alteraciones que reciben el nombre de fallas. Una falla en un circuito es cualquier evento que interfiere con el flujo normal de corriente [34].

Tipos de fallas

- Cortocircuitos: Trifásico simétrico, aislado o a tierra, bifásico aislado (cortocircuito entre 2 líneas), bifásico a tierra (entre dos líneas y el conjunto a tierra) y monofásico (una línea conectada a tierra).
- Fases abiertas: Una fase abierta, dos fases abiertas y tres fases abiertas. La última situación significa que la línea o dispositivo sale completamente de servicio.

Los cortocircuitos trifásicos dan origen a fallas simétricas pues el SEP permanece eléctricamente balanceado, en cambio los cortocircuitos bifásicos aislados y a tierra y el monofásico, así como 1 o 2 fases abiertas corresponden a fallas asimétricas, ya que el sistema queda eléctricamente desbalanceado en el punto de falla.

En el caso de fallas simétricas, el cálculo se realiza en base a una representación monofásica (por fase) de la red del SEP y se aplican las técnicas normales de análisis de circuitos. Para el cálculo de las fallas asimétricas, resulta conveniente utilizar al Método de las Componentes Simétricas [2].

Establecer una definición única de sistema de potencia, es algo difícil, ya que existe una gran cantidad de autores que han establecido sus puntos de vista al respecto.

Una de las definiciones más aceptadas a la escala mundial, es la establecida por el Institute of Electrical and Electronics Engineer (IEEE Instituto de Ingenieros Eléctricos y Electrónicos), esto define un sistema de Potencia como:

“Una red formada por unidades generadoras eléctricas, cargas y/o líneas de transmisión de potencia, incluyendo el equipo asociado, conectado eléctricamente o mecánicamente a la red” [3].

Por otra parte el diccionario de términos eléctricos y electrónicos de la IEEE, define el sistema de potencia como las fuentes de potencia eléctrica, conductores y equipos requerido para suplir la potencia eléctrica [36].

A continuación en la figura 1, se presenta una red eléctrica de potencia que se encarga de generar, transmitir y distribuir la energía eléctrica, hasta los distribuidores.

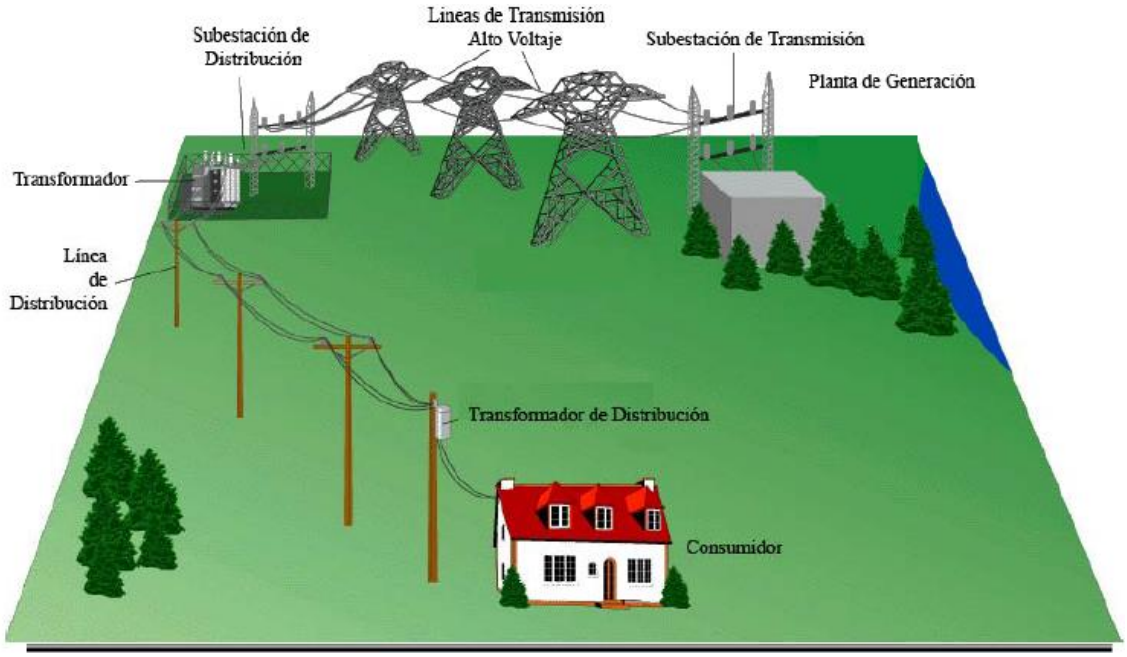

Figura 1 ***Estructura de un Sistema de Potencia***[4]

Hay varios métodos para resolver el FOP (Flujo Óptimo de Potencia), como son la programación lineal, el gradiente de base, la programación cuadrática entre otras [18].

No obstante, los métodos mencionados presentan tres problemas; en primer lugar no pueden proporcionar una solución óptima global sino un óptimo a nivel local, en segundo lugar, estos métodos se basan en hipótesis de continuidad y diferenciabilidad de la función objetivo y, por último, todos estos métodos no pueden aplicarse con variables discretas que es la variable que presenta separaciones o interrupciones en la escala de valores que puede tomar. Estas separaciones o interrupciones indican la ausencia de valores entre los distintos valores específicos que la variable pueda asumir. En este caso es para el valor del tap y potencia reactiva asignada a los capacitores/reactores conectados en derivación, se generan aleatoriamente entre sus límites máximos y mínimos.

Este trabajo presenta un algoritmo genético aplicado a la solución del problema de flujo óptimo de potencia (FOP), puesto que los algoritmos genéticos (AG) que son métodos sistemáticos para la resolución de problemas de búsqueda y optimización que aplican métodos de resolución efectivos como programas y software avanzados. Es decir son métodos adecuados para solucionar este problema, debido a que mejora los inconvenientes de los métodos convencionales de solución.

Con el estudio de los flujos de potencia se puede investigar lo siguiente:

Flujo en KW o VAR en las ramas de una red, voltajes en los buses, efecto de pérdidas temporales de generación o de circuitos de transmisión sobre las cargas del circuito, condiciones óptimas de operación del sistema de distribución de cargas, pérdidas optimas, influencia del cambio de tamaño de los conductores y posición óptima del cambiador de derivaciones de los transformadores [37].

1.2 Antecedentes

Antiguamente el estudio de los flujos de potencia se los realizaba en analizadores de redes de corriente alterna, los cuales suministraban simulación a pequeña escala y monofásica de la red real al interconectar los elementos del circuito y fuente de voltaje, el procedimiento de realizar las conexiones, los ajustes y tomar mediciones era tedioso y requería larga horas de tiempo, hoy en día una tabla de datos se puede obtener de manera inmediata con la facilidad que brindan los sistemas computacionales en la figura 1.2 vemos la lista desplegada de un circuito ejecutado por GAMS.

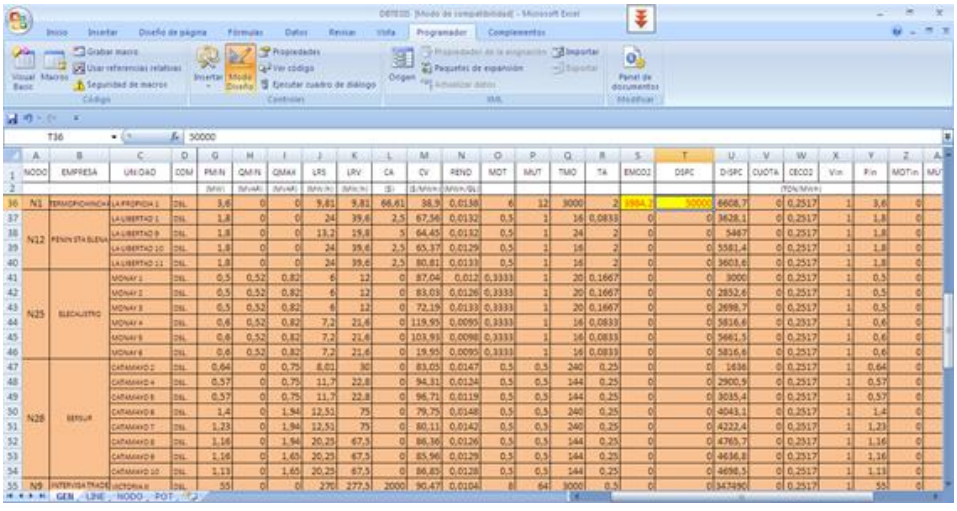

Figura 1.2 ***Presentación del menú principal y la base de datos*** [1]

[1] **Fuente:** El Autor

Gracias a ellos se puede lograr hallar las soluciones del estudio de los flujos de potencia de sistemas complejos, entre las innumerables ventajas se pueden manejar sistemas de más de 200 barras, 300 líneas y 500 transformadores.

Un ejemplo para obtener los valores de una manera matricial, como se muestra en la siguiente figura.

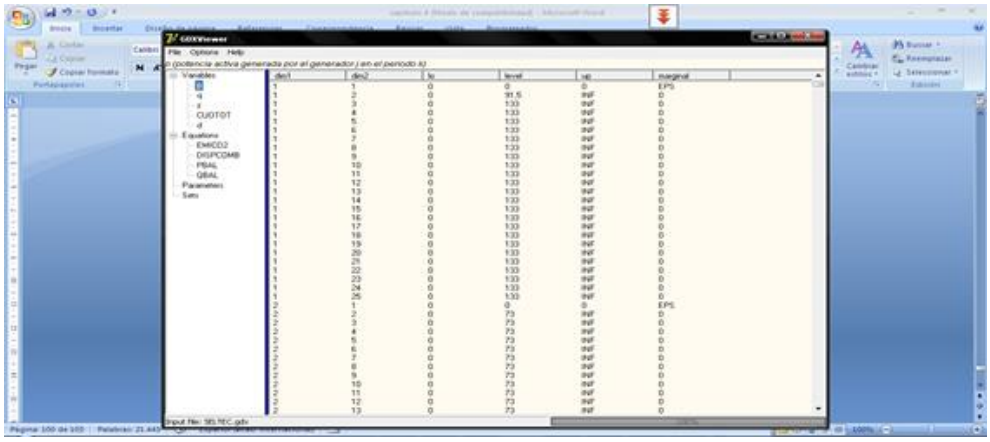

Figura 1.3 ***Aplicación GDXVIEWER***[2]

En la cual se puede visualizar los datos de potencias en filas y columnas enumeradas; y para que los datos sean exportados hacia EXCEL.

Los resultados completos son obtenidos de manera rápida y económica, simplemente con la impresión de los reportes. Los listados impresos de los resultados que provee la computadora consisten en una serie de tablas.

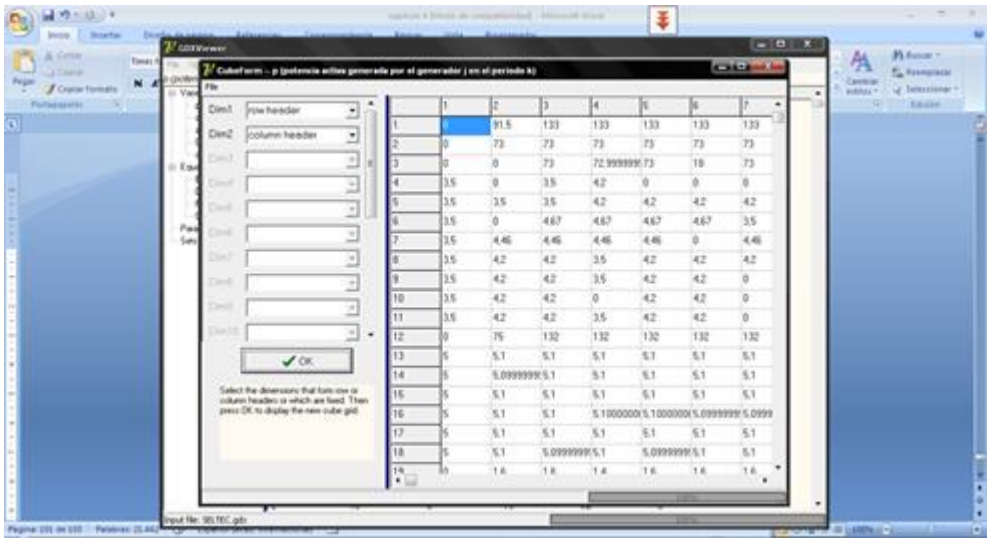

Figura 1.4 ***Exportar mediante tablas hacia EXCEL***[3]

[2 y 3] **Fuente:** El Autor

1.3 Ecuaciones de flujos de potencia

Gran parte de los métodos para resolver el problema de flujos de potencia se basa en las ecuaciones nodales de la red [39].

La forma de las ecuaciones nodales de un sistema $n'1$ nodos mayores uno de los cuales, el neutro, se tome como referencia para los voltajes, es la siguiente:

$$\check{Y}_{11}\check{V}_1 + \check{Y}_{13}\check{V}_2 + \ldots\ldots + \check{Y}_{1k}\check{V}_k + \ldots\ldots\ldots + \check{Y}_{1n}\check{V}_n = \check{I}_1$$
$$\check{Y}_{21}\check{V}_1 + \check{Y}_{22}\check{V}_2 + \ldots\ldots + \check{Y}_{2k}\check{V}_k + \ldots\ldots\ldots + \check{Y}_{2n}\check{V}_n = \check{I}_2$$
$$\check{Y}_{k1}\check{V}_1 + \check{Y}_{k2}\check{V}_2 + \ldots\ldots + \check{Y}_{kk}\check{V}_k + \ldots\ldots\ldots + \check{Y}_{kn}\check{V}_n = \check{I}_k$$
$$\check{Y}_{n1}\check{V}_1 + \check{Y}_{n2}\check{V}_2 + \ldots\ldots + \check{Y}_{nk}\check{V}_k + \ldots\ldots\ldots + \check{Y}_{nm}\check{V}_m = \check{I}_n$$

Donde $\check{Y}$ = admitancia
$\check{V}$ = voltaje
$\check{I}$ = corriente

Las fuentes de corriente que se muestran en las ecuaciones anteriores, que representan los generadores y las cargas pueden expresarse con función de la potencia real y reactiva en par unidad, inyectadas o sustraídas en cada punto de unión [19].

Por ejemplo:

$$\check{I}_k = \left(\frac{\check{P}_k\, j\, \check{Q}_k}{\check{V}_k}\right) = \frac{\check{P}_k - j\check{Q}_k}{\check{V}_k}$$

Siendo $\check{Q}$ = potencia reactiva
$\check{P}$ = potencia real
$\overline{\check{V}}$ = voltaje
$\check{I}$ = corriente

Y la ecuación quedara:

$$\check{Y}_{k1}\check{V}_1 + \check{Y}_{k2}\check{V}_2 + \ldots\ldots + \check{Y}_{kk}\check{V}_k + \ldots\ldots\ldots + \check{Y}_{kn}\check{V}_n = \frac{\check{P}_k\, j\, \check{Q}_k}{\overline{\check{V}^0}_k} \qquad (1)$$

En las barras de carga donde se conocen la potencia real y reactiva, la ecuación puede plantearse directamente. En las barras de generación, donde se especifica la potencia real generada y el módulo del voltaje terminal del generador, es conveniente expresar la potencia reactiva en función de los voltajes y las admitancias de la red [17].

El problema consiste en determinar los voltajes, en módulos y argumento en todas las barras resolviendo el sistema de n ecuaciones simultaneas de la forma de la ecuación (1).

Este es un sistema de ecuaciones no lineales, por lo que se recurre a métodos interactivos para obtener la solución. Los métodos más utilizados son el de Gauss Seidal y el Newton Raphson [29].

Método Gauss Seidal

Se considera el método de eliminación para resolver ecuaciones simultáneas suministra soluciones suficientemente precisas hasta para 15 o 20 ecuaciones. El número exacto depende de las ecuaciones de que se trate, del número de dígitos que se conservan en el resultado de las operaciones aritméticas, y del procedimiento de redondeo. Utilizando ecuaciones de error, el número de ecuaciones que se pueden manejar se puede incrementar considerablemente a más de 15 o 20, pero este método también es impráctico cuando se presentan, por ejemplo, cientos de ecuaciones que se deben resolver simultáneamente. El método de inversión de matrices tiene limitaciones similares cuando se trabaja con números muy grandes de ecuaciones simultáneas [20].

Sin embargo, existen varias técnicas que se pueden utilizar, para resolver grandes números de ecuaciones simultáneas. Una de las técnicas más útiles es el método de Gauss-Seidel. Ninguno de los procedimientos alternos es totalmente satisfactorio, y el método de Gauss-Seidel tiene la desventaja de que no siempre converge a una solución o de que a veces converge muy lentamente. Sin embargo, este método convergirá siempre a una solución cuando la magnitud del coeficiente de una incógnita diferente en cada ecuación del conjunto, sea suficientemente dominante con respecto a las magnitudes de los otros coeficientes de esa ecuación.

Es difícil definir el margen mínimo por el que ese coeficiente debe dominar a los otros para asegurar la convergencia y es aún más difícil predecir la velocidad de la convergencia para alguna combinación de valores de los coeficientes cuando esa convergencia existe. No obstante, cuando el valor absoluto del coeficiente dominante para una incógnita diferente para cada ecuación es mayor que la suma de los valores absolutos de los otros coeficientes de esa ecuación, la convergencia está asegurada. Ese conjunto de ecuaciones simultáneas lineales se conoce como sistema diagonal.

Un sistema diagonal es condición suficiente para asegurar la convergencia pero no es condición necesaria. Afortunadamente, las ecuaciones simultáneas lineales que se derivan de muchos problemas de ingeniería, son del tipo en el cual existen siempre coeficientes dominantes [27].

La secuencia de pasos que constituyen el método de Gauss-Seidel es la siguiente:

1. Asignar un valor inicial a cada incógnita que aparezca en el conjunto. Si es posible hacer una hipótesis razonable de éstos valores, hacerla. Si no, se pueden asignar valores seleccionados arbitrariamente. Los valores iniciales utilizados no afectarán la convergencia como tal, pero afectarán el número de iteraciones requeridas para dicha convergencia.

2. Partiendo de la primera ecuación, determinar un nuevo valor para la incógnita que tiene el coeficiente más grande en esa ecuación, utilizando para las otras incógnitas los valores supuestos.

3. Pasar a la segunda ecuación y determinar en ella el valor de la incógnita que tiene el coeficiente más grande en esa ecuación, utilizando el valor calculado para la incógnita del paso 2 y los valores supuestos para las incógnitas restantes.

4. Continuar con las ecuaciones restantes, determinando siempre el valor calculado de la incógnita que tiene el coeficiente más grande en cada ecuación particular, y utilizando siempre los últimos valores calculados para las otras incógnitas de la ecuación. (Durante la primera iteración, se deben utilizar los valores supuestos para las incógnitas hasta que se obtenga un valor calculado). Cuando la ecuación final ha sido resuelta, proporcionando un valor para la única incógnita, se dice que se ha completado una iteración.

5. Continuar iterando hasta que el valor de cada incógnita, determinado en una iteración particular, difiera del valor obtenido en la iteración previa, en una cantidad menor que cierto EPSILON seleccionado arbitrariamente. El procedimiento queda entonces completo.

Refiriéndonos al paso 5, mientras menor sea la magnitud del EPSILON seleccionado, mayor será la precisión de la solución. Sin embargo, la magnitud del EPSILON no especifica el error que puede existir en los valores obtenidos para las incógnitas, ya que ésta es una función de la velocidad de convergencia. Mientras mayor sea la velocidad de convergencia, mayor será la precisión obtenida en los valores de las incógnitas para un EPSILON dado.

Método Newton Raphson

Este método, es un método iterativo, es uno de los más usados y efectivos. A diferencia de los métodos anteriores, el método de Newton Raphson no trabaja sobre un intervalo sino que basa su fórmula en un proceso iterativo [25].

Supongamos que tenemos la aproximación x_i a la raíz x_y de $f(x)$,

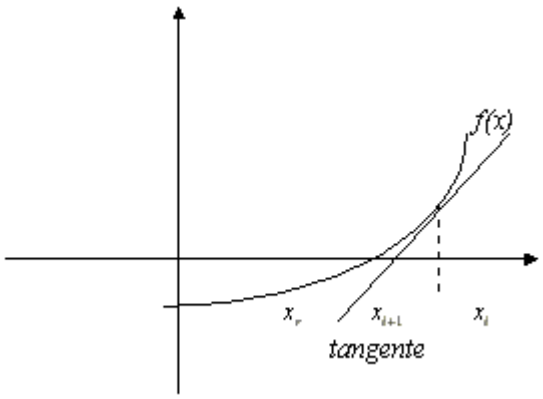

Figura *1.5 Aproximación xi a la raíz xy de f(x)* [59].

Trazamos la recta tangente a la curva en el punto $(x_i, f(x_i))$; ésta cruza al eje x en un punto x_{i+1} que será nuestra siguiente aproximación a la raíz x_y [57].

Para calcular el punto x_{i+1} , calculamos primero la ecuación de la recta tangente. Sabemos que tiene pendiente

$$m = f'(x_i)$$

Y por lo tanto la ecuación de la recta tangente es:

$$y - f(x_i) = f'(x_i)(x - x_i)$$

Hacemos $y = 0$

$$-f(x_i) = f'(x_i)(x - x_i)$$

Y despejamos x :

$$x = x_i - \frac{f(x_i)}{f'(x_i)}$$

Que es la fórmula iterativa de Newton-Raphson para calcular la siguiente aproximación:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}, \; si \; f'(x_i) \neq 0$$

Note que el método de Newton Raphson no trabaja con intervalos donde nos asegure que encontraremos la raíz, y de hecho no tenemos ninguna garantía de que nos aproximaremos a dicha raíz. Desde luego, existen ejemplos donde este método no converge a la raíz, en cuyo caso se dice que el método diverge. Sin embargo, en los casos donde si converge a la raíz lo hace con una rapidez impresionante, por lo cual es uno de los métodos preferidos por excelencia.

También observe que en el caso de que $f'(x_i) = 0$, el método no se puede aplicar. De hecho, vemos geométricamente que esto significa que la recta tangente es horizontal y por lo tanto no intersecta al eje x en ningún punto, a menos que coincida con éste, en cuyo caso x_i mismo es una raíz de $f(x)$!

Una vez conocidos todos los voltajes de los nodos pueden calcularse los flujos de corrientes en todas las ramas de la red, cuyas admitancias son conocidas y los flujos de potencia real y reactiva.

1.3.1 Solución de las ecuaciones de flujo de potencia por el método de Gauss Seidel

Despejando de la ecuación (1) $\check{V}_k$

$$\check{V}_k = \frac{1}{\check{Y}_{kk}} \left[\frac{\check{P}_k - j\,\check{Q}_k}{\check{V}_k} - \sum_{m=1}^{n} \check{Y}_{kn}\,\check{V}_m \right] \qquad (2)$$

Donde $m \neq k$

El cálculo puede hacerse de la siguiente manera:

Partiendo de los valores supuestos de los n voltajes se calcula el valor de voltaje en el punto de unión 1, mediante la ecuación (2) para $k = 1$

$$\check{V}_1 = \frac{1}{\check{Y}_{11}} \left[\check{P}_1 - j\check{Q}_1 - \sum_{m=2}^{n} \check{Y}_{km}\check{V}_n \right] \qquad (3)$$

El valor corregido del voltaje de punto de unión 1 se utiliza para calcular en forma similar el valor corregido del voltaje de punto de unión 2. El proceso se repite en cada barra, hasta incluir las n barras, lo que contempla la primera iteración. Con los valores de los n voltajes obtenidos en la primera iteración, se repite el proceso todas las veces que sea necesario, hasta que la diferencia entre los valores de los voltajes de cada barra calculados en dos iteraciones sucesivas sea menos que una tolerancia predeterminada.

En las barras de generación donde se conocen la potencia real y el módulo del voltaje, debe calcularse la potencia reactiva mediante la expresión que se deduce a continuación:

De la ecuación (1)

$$\check{P}_k - j\check{Q}_k = -\check{V}_k^{\,*} \sum_{m=1}^{n} \check{Y}_{km}\check{V}_m$$

La parte imaginaria de la expresión anterior con signo negativo corresponde a la potencia reactiva

$$\check{Q}_k = -\delta_m \left[\check{V}_k^{\,*} \sum_{m=1}^{n} \check{Y}_{km}\check{V}_m \right] \qquad (4)$$

El método iterativo de Gauss – Seidel calcula la potencia reactiva Q_k mediante la ecuación (3), partiendo de los valores disponibles de los voltajes y se sustituye en la ecuación (2) para encontrar una nueva aproximación del voltaje V_k.

1.3.2 Solución de las ecuaciones de flujo de potencia por el método de Newton Raphson.

La potencia compleja generada o substraída de una barra cualquiera k de un sistema de n barras, puede expresarse, tomando como base la ecuación (1) de la forma siguiente:

$$\check{P}_k - j\check{Q}_k = \check{V}_k^{\,*} \sum_{m=1}^{n} \check{Y}_{km}\check{V}_m \qquad (5)$$

Los voltajes y las admitancias se pueden expresar, usando coordenadas rectangulares:

$$\check{V}_k = \check{e}_k + j\check{f}_k$$
$$\check{V}_m = \check{e}_m + j\check{f}_m$$
$$\check{Y} = \check{G}_{km} - j\,\check{B}_{km}$$

Sustituyendo en la ecuación (4)

$$\check{P}_k - j\check{Q}_k = \left|e_k - j\check{f}_k\right| \sum_{m-1}^{n} \left[\check{G}_{km} - j\,\check{B}_{km}\right]\left[\check{e}_m + j\check{f}_m\right]$$

La potencia real o reactiva P_k es igual a la parte real de la expresión anterior y la potencia reactiva Q_k. Es igual a la parte imaginaria multiplicada por -1.

$$\check{P}_k = \sum_{m-1}^{n} \left[\check{e}_k\left(\check{e}_m\check{K}_{km} + \check{f}_m\check{B}_{km}\right) + \check{f}_k\left(\check{f}_m\check{G}_{km} - \check{e}_m\check{B}_{km}\right)\right] \qquad (6)$$

$$\check{Q}_k = \sum_{m-1}^{n} \left[\check{f}_k\left(\check{e}_m\check{K}_{km} + \check{f}_m\check{B}_{km}\right) - \check{e}_k\left(\check{f}_m\check{G}_{km} - \check{e}_m\check{B}_{km}\right)\right] \qquad (7)$$

Donde $\check{P}_k =$ Potencia real
$\check{Q}_k =$ Potencia reactiva

De aquí se resolverá dos ecuaciones simultáneas no lineales para cada barra, de tal forma que si el sistema tiene n barras resulta un sistema de $2n$ ecuaciones. Se tiene luego un total de $2n$ incógnitas, 2 por barra, de la siguiente manera:

a.- En las barras de carga, donde se señala la potencia real y reactiva sustraídas, las incógnitas son el modulo y el argumento del voltaje de la barra.

b.- En las barras de generación donde se señala la potencia real producida or el generador y el modulo del voltaje de la barra, las incógnitas son la potencia reactiva suministrada por el generador y el ángulo del voltaje.

c.- En una barra de generación en la que se especifica el modulo y el argumento del voltaje, las incógnitas son la potencia real y potencia reactiva suministrado por el generador.

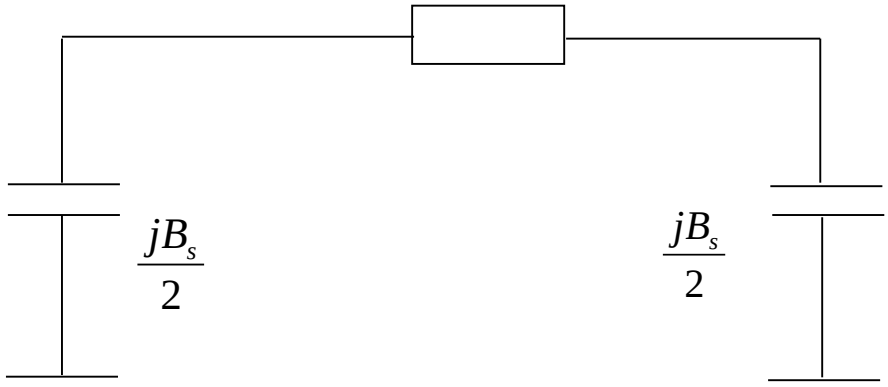

Figura 1.6 ***Modelo de línea de trasmisión***

A continuación supongamos que se tiene un sistema de tres ecuaciones simultáneas no lineales.

$$Y_1 = f_1(x_1, x_2, x_3)$$
$$Y_2 = f_2(x_1, x_2, x_3)$$
$$Y_3 = f_3(x_1, x_2, x_3)$$

Donde x_1, x_2, x_3 n incógnitas
$Y =$ Admitancias

Si se conoce los valores de Y_1, Y_2 y Y_3 y se deben calcular de x_1, x_2 y x_3 que satisfacen las ecuaciones. Hacemos luego un cálculo inicial de incógnitas, esos calores iniciales se representan con los símbolos.

$$x^\circ_1, x^\circ_2, x^\circ_3$$

Esta primera aproximación no satisface las ecuaciones. Entonces llamamos $\Delta x^\circ_1, \Delta x^\circ_2$ y Δx°_3 a las cantidades que hay que sumarle a los valores inicialmente supuestos de las variables para que el sistema de ecuaciones se verifique. Por lo tanto puede escribirse:

$$Y_1 = f_1\left(x^\circ_1 + \Delta x^\circ_1, x^\circ_2 + \Delta x^\circ_2, x^\circ_3 + \Delta x^\circ_3,\right)$$
$$Y_2 = f_2\left(x^\circ_1 + \Delta x^\circ_1, x^\circ_2 + \Delta x^\circ_2, x^\circ_3 + \Delta x^\circ_3,\right)$$
$$Y_3 = f_3\left(x^\circ_1 + \Delta x^\circ_1, x^\circ_2 + \Delta x^\circ_2, x^\circ_3 + \Delta x^\circ_3,\right)$$

Se sabe que cualquier función de x que tenga derivadas de todas las órdenes en el punto $x = x_1$, puede expresarse como una serie de Taylor, de la siguiente forma:

$$f(x) = f(x_1) + f'(x_1)(x - x_1) + \frac{f''(x_1)}{2!}(x - x_1)^2 \ldots\ldots + \frac{f''(x_1)}{n!}(x - x_1)^n$$

Aplicando la expansión en una serie de Taylor al caso de tres ecuaciones simultaneas con funciones de tres variables, tomando los dos primeros términos de la serie y despreciando los otros. El error será mínimo si la primera estimación de las variables las siguientes ecuaciones:

$$Y_1 = f_1(x^\circ_1, x^\circ_2, x^\circ_3) + \frac{\delta f_1}{\delta x_1}\left|\Delta x^\circ_1 + \frac{\delta f_1}{\delta x_2}\right|_0 \Delta x^\circ_2 + \frac{\delta f_1}{\delta x_3}\left|\Delta x^\circ_2\right.$$
$$Y_2 = f_2(x^\circ_1, x^\circ_2, x^\circ_3) + \frac{\delta f_2}{\delta x_1}\left|\Delta x^\circ_1 + \frac{\delta f_2}{\delta x_2}\right|_0 \Delta x^\circ_2 + \frac{\delta f_2}{\delta x_3}\left|\Delta x^\circ_3\right.$$
$$Y_3 = f_3(x^\circ_1, x^\circ_2, x^\circ_3) + \frac{\delta f_3}{\delta x_1}\left|\Delta x^\circ_1 + \frac{\delta f_3}{\delta x_2}\right|_0 \Delta x^\circ_2 + \frac{\delta f_1}{\delta x_3}\left|\Delta x^\circ_3\right.$$

Las derivadas parciales se evalúan para la aproximación de las incógnitas o sea para x°_1, x°_2 y x°_3 respectivamente. Luego se puede utilizar la notación matricial.

1.3.3 Solución aproximada de los flujos de potencia.

Según la ecuación

$$\check{P}_1 - j\check{Q}_k = \bar{\bar{V}}_k - \sum_{m=2}^{n} \check{Y}_{km}\bar{\bar{V}}_m$$

Expresando los voltajes y admitancias en coordenadas polares

$$\bar{\bar{V}}_m = \check{V}_k\, e^{-j\delta k} \qquad \check{V}_k = \check{V}_k\, e^{-j\delta k} \qquad \bar{\bar{V}}_m = \check{V}_m\, e^{-j\delta k}$$
$$\check{V}_{km} = |\check{V}_{km}|e^{-j\theta km}$$

Sustituyendo los valores en la ecuación (5), se tiene:

$$\check{P}_k - j\check{Q}_k = \sum_{m=1}^{n}\left(\check{V}_k\, \check{V}_m\, \check{Y}_{km}\right)e^{-j\,(\delta_k - \delta_m + \theta_{km})} \qquad (11)$$

Teniendo en cuenta que:

$$e^{-j\,(\delta_k - \delta_m + \theta_{km})} = \cos(\delta_k - \delta_m + \theta_{km}) - \operatorname{sin}(\delta_k - \delta_m + \theta_{km})$$

La potencia real o activa y la potencia reactiva pueden expresarse como la parte real y la parte imaginaria respectivamente (11):

$$\check{P}_k = \sum_{m=1}^{n}\left(\check{V}_k\, \check{V}_m\, \check{Y}_{km}\right)\cos(\delta_k - \delta_m + \theta_{km}) \qquad (12)$$

$$\check{Q}_k = \sum_{m=1}^{n}\left(\check{V}_k\, \check{V}_m\, \check{Y}_{km}\right)\operatorname{sen}(\delta_k - \delta_m + \theta_{km}) \qquad (13)$$

Ecuaciones que relacionan cambios de potencia real y reactiva en función de cambios de voltaje se escriben en forma abreviada:

$$\begin{bmatrix}\Delta P \\ \Delta Q\end{bmatrix} = \begin{bmatrix}\check{J}_1 & \check{J}_2 \\ \check{J}_3 & \check{J}_4\end{bmatrix}\begin{bmatrix}\Delta\delta \\ \Delta P\end{bmatrix} \qquad (14)$$

Donde $\Delta\delta$ y $\Delta\check{V}$ son incrementos y decrementos de los ángulos y de los módulos de voltaje de barras. Los elementos de la matriz Jacobiana se calculan obteniendo las derivadas parciales de las ecuaciones (12) *y* (13) para la submatriz $\check{J}_1$ [61].

Las variaciones de la potencia real se deben principalmente a las variaciones del ángulo de voltaje y las variaciones de la potencia reactiva se deben a las variaciones en el módulo del voltaje. Para cambios pequeños de magnitud y el ángulo de los voltajes, las submatrices se pueden considerar cero y la ecuación matricial (7) se reduce a:

$$\begin{bmatrix}\Delta P\\ \Delta Q\end{bmatrix} = \begin{bmatrix}\check{J}_1 & 0\\ 0 & \check{J}_4\end{bmatrix}\begin{bmatrix}\Delta\delta\\ \Delta P\end{bmatrix} \qquad (15)$$

Lo que significa que puedes calcularse separadamente los flujos de potencia real y activa:

$$[\Delta P] = [\check{J}_1]\,[\Delta\delta]$$
$$[\Delta Q] = [\check{J}_4]\,[\Delta V]$$

Esta simplificación se llama de los flujos desacoplados[26].

1.4 Conceptos Básicos

1.4.1 Potencia eléctrica, activa y reactiva.

La definición de potencia en términos de energía es "la cantidad de energía consumida o generada por unidad de tiempo". Para el caso particular de potencia eléctrica, se establece la definición: "la potencia eléctrica generada o absorbida por un elemento es el producto del voltaje en sus terminales y la corriente a través de él" [24], algebraicamente está dada por:

$$p = vi \qquad \text{Watt o Joule/seg} \qquad (1.1)$$

Donde p = Potencia Real
v = Voltaje Real
i = Corriente

Una vez que se ha definido la potencia eléctrica, es interesante analizar cómo es consumida por los elementos pasivos.

Por ejemplo para el caso de una resistencia a la cual se le aplica una señal del tipo alterna, es decir $v = V_m \sin wt$, por lo que la respuesta de este elemento ante una señal alterna es $v = V_m \sin wt$, $i = I_m \sin wt$, por lo tanto sustituyendo en (1.1) se tiene:

$$P_R = V_m I_m \sin^2 wt \qquad (1.2)$$

Se observa que la potencia eléctrica consumida por una resistencia es positiva, aunque tenga una variación en el tiempo como lo muestra la expresión (1.2). En la Figura 1.7 se tiene gráficamente la variación de la potencia eléctrica consumida por la resistencia al aplicarle una señal de corriente alterna

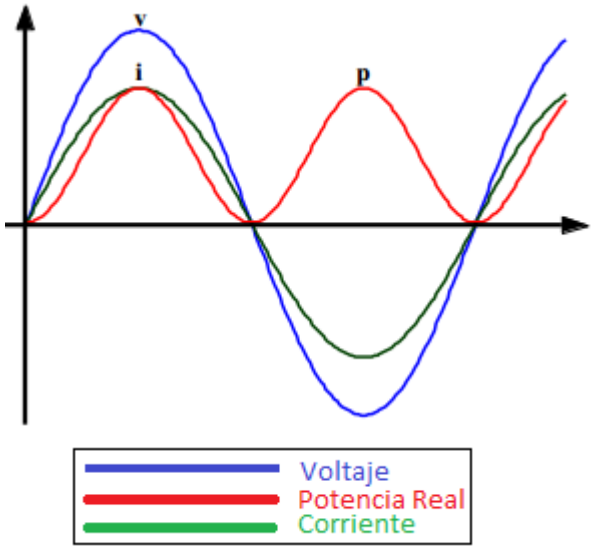

Figura 1.7 ***Variación con respecto al tiempo de v, i y p para una resistencia***

Cuando la carga en una fuente de CA contiene solamente resistencia, la corriente y el voltaje están en fase, esto significa que dos formas de onda llegan juntas al valor de pico positivo, a cero y el valor de pico negativo. La corriente en la figura 1.7 está en fase con el voltaje. En cada instante la potencia es igual a la corriente en este instante multiplicada por el voltaje en ese instante. Como se ve el producto de los valores instantáneos de las corrientes y voltajes crean una potencia instantánea. Representando todas las potencias instantáneas obtenemos una forma de onda de la potencia. Sin embargo, obsérvese que la forma de onda de potencia siempre es positiva para corrientes y voltajes en fase.

De igual forma se aplica una señal de voltaje de corriente alterna a un inductor de la forma $v = V_m \sin\omega t$, obteniéndose como respuesta una corriente a través de él del tipo

$i = -I_m \cos \omega t$, recordando que la relación entre voltaje y corriente es $v = L\frac{di}{dt}$, por lo tanto la potencia instantánea a través del elemento se expresa mediante la ecuación (1.3). La Figura 1.3 muestra gráficamente las variables eléctricas de un inductor ante una excitación senoidal.

$$P_L = -V_m I_m \operatorname{sin}\omega\, t \cos \omega t \qquad (1.3)$$

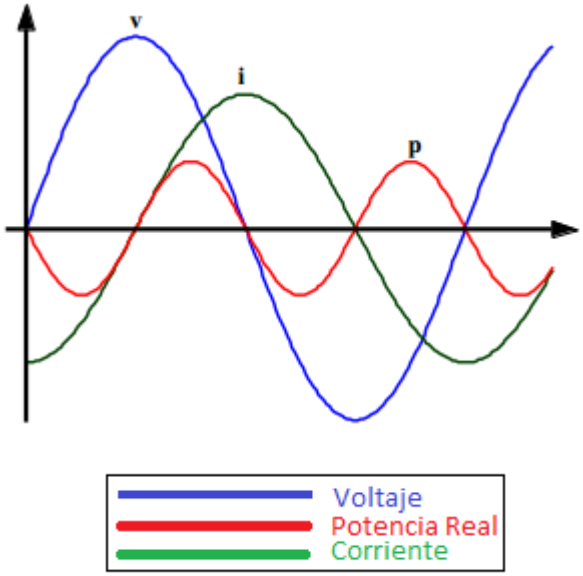

Figura 1.8 ***Variación con respecto al tiempo de v, i y p para una inductancia***

Las bobinas de autoinducción producen reactancias inductivas. El efecto de la reactancia inductiva es hacer que la corriente se retrase del voltaje. En la figura 1.8 el voltaje ya ha alcanzado su valor de pico positivo, mientras que la corriente solamente está empezando a ser positiva. Las cargas que contengas solamente capacidad o solamente inductancia, hacen que la corriente y el voltaje estén desfasados en 90 grados. Además, toma valores positivos y negativos con amplitudes máximas iguales lo que lleva a concluir que la onda de potencia instantánea tiene un valor promedio cero

Caso similar ocurre cuando se le aplica en terminales de un capacitor un voltaje $v = V_m \operatorname{sin} \omega t$, circulando a través del elemento una corriente de la forma $i = I_m \cos \omega t$, la potencia instantánea es el producto de estas dos señales, por lo que se llega a la expresión (1.4) la Figura 1.9 presenta en forma gráfica las señales eléctricas de un capacitor.

$$P_C = V_m I_m \sin wt \cos wt \quad (1.4)$$

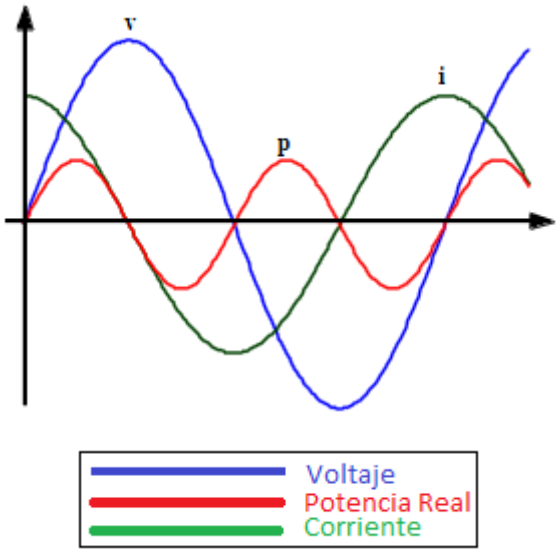

Figura 1.9 ***Variación con respecto al tiempo de v, i y p para una capacitancia***

De las gráficas anteriores se observa que la potencia suministrada a un elemento puramente inductivo o capacitivo es absorbida durante un cuarto de la onda de voltaje y devuelta a la fuente durante el siguiente cuarto de la onda. Se puede decir que la potencia en estos dos elementos tiene un comportamiento reactivo, por lo que puede decirse que es una potencia reactiva. A diferencia de la potencia en un elemento puramente resistivo en el cual siempre es positiva, por lo que puede considerarse como una potencia activa.

Si ahora se analiza el comportamiento de la potencia eléctrica instantánea en un circuito más general, es decir, uno que contenga resistencia, inductancia y capacitancia como se muestra en la Figura 1.10, al cual se le energiza con una señal de voltaje alterna del tipo $v = V_m \sin \omega t$, obteniéndose una respuesta también alterna de la forma $i = I_m \sin(\omega t + \emptyset)$,

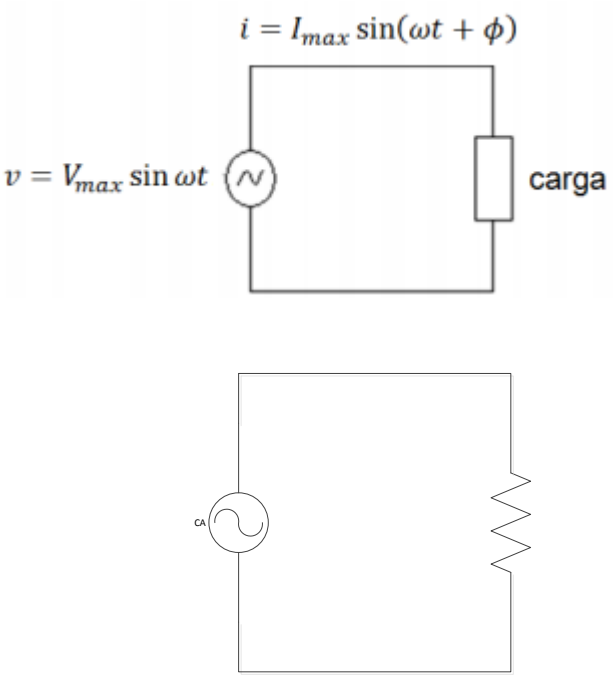

Figura 1.10 ***Circuito eléctrico con elementos R, L y C.***

Dibujamos el diagrama de vectores teniendo en cuenta:

Que la intensidad que pasa por todos los elementos es la misma, que la suma (vectorial) de las diferencias de potencial entre los extremos de los tres elementos nos da la diferencia de potencial en el generador de corriente alterna.

La potencia eléctrica en el circuito será entonces:

$$p = V_m I_m \sin\omega t \sin(\omega t + \emptyset) \qquad (1.5)$$

Utilizando identidades trigonométricas y manipulando la ecuación anterior puede rescribirse como:

$$p = \frac{V_m I_m}{2}[\cos\emptyset\,(1 - \cos 2\,\omega t) + \sin\emptyset\,\sin 2\,\omega t] \qquad (1.6)$$

La potencia instantánea se descompone en dos términos; recordando que los valores máximos pueden ser expresados como valores eficaces utilizando la relación $|V| = \frac{V_m}{\sqrt{2}}$ por lo tanto se tiene:

$$p = |V|\,|I|\,\cos\emptyset\,(1 - \cos 2\,\omega t) + |V|\,|I|\sin\emptyset\,\sin 2\,\omega t \qquad (1.7)$$

En la ecuación (1.7) se observa que la potencia instantánea oscila alrededor de un valor promedio dado por el primer término de la expresión, con la particularidad de que nunca se hace negativa, mientras que el segundo término tiene un valor promedio cero.

Definiendo entonces las siguientes cantidades:

$P = |V|\,|I|\,\cos\emptyset$ Potencia activa

$Q = |V|\,|I|\,\sin\emptyset$ Potenca reactiva (1.8)

Sustituyeron (1.8) en (1.7) se simplifica la expresión:

$$p = \mathrm{P}\,(1 - \cos 2\,\omega t\,) + \mathrm{Q}\sin 2\,\omega t \qquad (1.9)$$

En la Figura 1.11 se tiene la variación de la potencia instantánea con respecto al tiempo, así como las variaciones voltaje y corriente para el circuito de la Figura 1.10

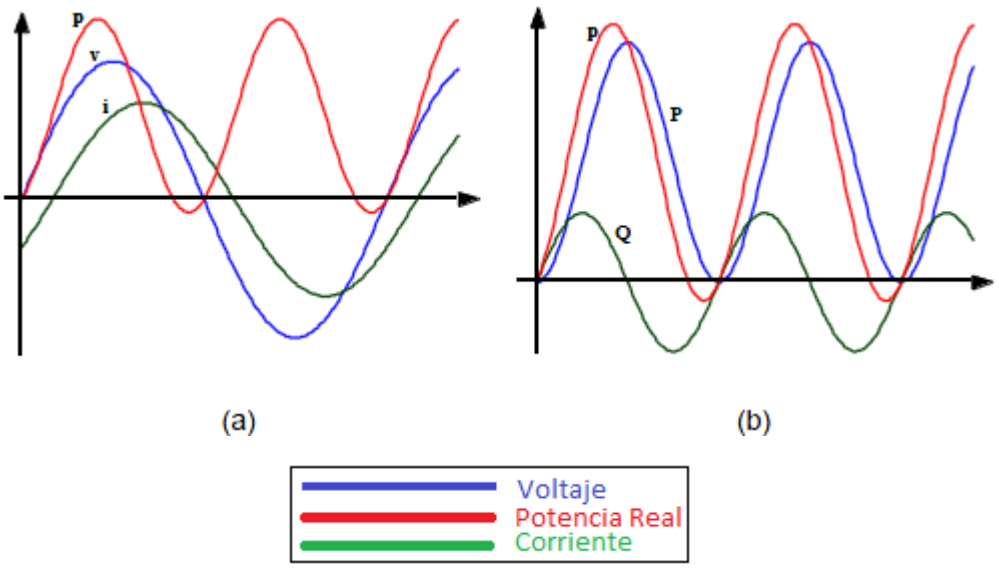

Figura 1.11 ***Variación con respecto al tiempo de v, i y p para un circuito RLC.***

Para los circuitos en los que ocurre desfase como en RLC el cálculo de la potencia no es solamente multiplicar la corriente y el voltaje, en la figura 1.11 se muestra casos de desfase y de potencia resultante, obsérvese que en algunos casos la corriente y el voltaje son de polaridades opuestas, Esto conduce a una potencia negativa para esta parte del ciclo, esto significa que la carga no toma potencia de la fuente durante este intervalo de tiempo.

En la Figura 1.11 (a) y (b) se observa que la potencia instantánea toma valore negativos durante ciertos periodos de tiempo, indicando con esto que la energía fluye en esos momentos de la carga al generador.

De las expresiones y gráficas anteriores se puede concluir que la Potencia Activa se define como el valor promedio alrededor del cual oscila la potencia instantánea, por lo que representa la potencia útil, aquella que es capaz de realizar un trabajo o que se disipa en forma de calor. Mientras que la Potencia Reactiva se define como el valor pico de una de las componentes de la potencia instantánea, cuyo valor promedio es cero y que por lo tanto no es capaz de realizar trabajo útil, pero que se desplaza continuamente del generador a la carga y viceversa.

1.4.2 Potencia Compleja.

Para facilitar el análisis de comportamiento de redes eléctricas en régimen permanente, cuando estas son excitadas por señales de tipo alterno, se desarrolla una transformación denominada fasorial, mediante la cual una función del tipo senoidal puede representarse por un número complejo denominado fasor. Considerando el circuito eléctrico elemental mostrado en la siguiente figura:

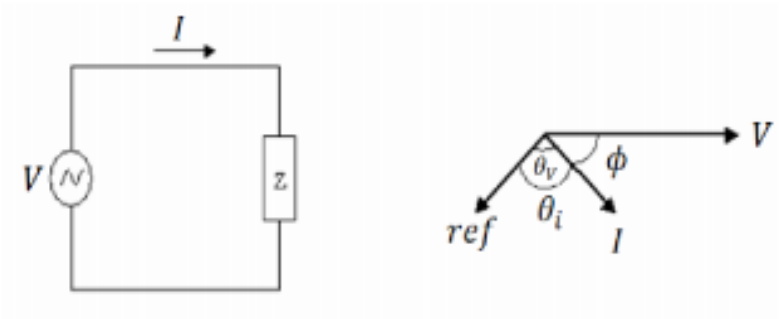

Figura *1.12 Circuito eléctrico monofásico.*

El voltaje y la corriente del circuito se pueden expresar en forma fasorial como:

$$V = |V|e^{j\theta v} = |V|(\cos\theta v + j\sin\theta v)$$

$$I = |I|e^{j\theta i} = |I|(\cos\theta i + j\sin\theta i) \qquad (1.10)$$

De acuerdo con la condición original de potencia instantánea dada por $p = vi$, la potencia compleja se define como:

$$S = VI^* = |V|\, e^{j\theta v}\, |I|e^{-j\theta i} = |V||I|e^{j(\theta v - \theta i)} \qquad (1.11)$$

En la expresión anterior se introduce un concepto que se conoce como potencia aparente y se simboliza por la letra S. Además, de la misma expresión, el ángulo $(\theta_v - \theta_i)$ es el ángulo de desfasamiento entre el voltaje y la corriente $(\emptyset)$.
Por lo que (1.11) la formula anterior se puede escribir como:

$$p = |V|\,|I| \cos\emptyset + \mathrm{j}\,|V|\,|I| \sin\emptyset$$
$$S = P + jQ \qquad (1.12)$$

1.4.3 Triángulo de potencias.

La relación que existe entre potencia aparente, reactiva y activa puede ser visto en forma gráfica utilizando lo que se conoce como triángulo de potencia, por lo que si comparamos las componentes de la intensidad, activa y reactiva, respecto a la tensión, en circuitos inductivos y capacitivos, y multiplicamos por la tensión, los triángulos formados por la intensidad y componentes obtenemos la figura 1.13 que constituirán los denominados triángulos de potencia el cual se muestra en la siguiente Figura:

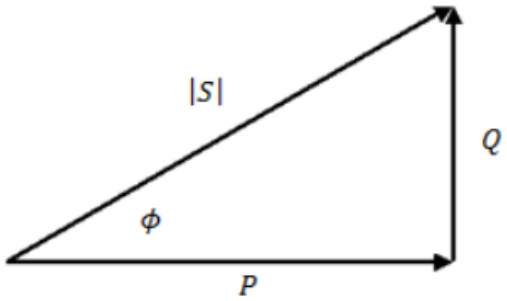

Figura 1.13 ***Triángulo de potencia*****.**

Del triángulo de potencia se obtienen las expresiones:

$$S = V\,I^* = P + jQ$$
$$|V| = \sqrt{P^2 + Q^2}$$
$$\cos\theta = \frac{P}{S}$$

En donde $\emptyset$ representa una medida de la cantidad de potencia útil que está siendo consumida por el elemento, por lo que al $\cos \emptyset$ se le conoce como factor de potencia, el cual al multiplicarlo por la potencia aparente, resulta en la potencia activa que el elemento consume.

1.4.4 Sistema Por Unidad

Cuando se estudian sistemas eléctricos de potencia balanceados, el uso de los valores reales de los elementos (Ω, A, V, VA) hace más complejo el análisis que si se utilizan sus valores en por unidad (p.u.). El uso de valores en p.u. es muy común entre quienes realizan estudios de sistemas eléctricos de potencia y los fabricantes prefieren especificar las impedancias y reactancias de sus generadores y transformadores en esta misma forma.

Algunas de las razones por las que se considera apropiado trabajar en por unidad son las siguientes:

1. Las cantidades involucradas en el proceso de solución de problemas de sistemas de potencia son de gran magnitud: Kilovoltios KV, Kiloamperios KA, Megavoltamperios MVA. Estas cantidades requieren más espacio de memoria en los sistemas de cómputo para ser almacenadas, su manipulación en los procesos consume mayor tiempo de máquina y se aumenta la posibilidad de producir y propagar errores numéricos.
2. Si se utiliza un valor base o de referencia para especificar las cantidades de interés en el análisis del sistema, la observación del estado operativo bueno o malo es más directo que si se utilizan valores reales, ya que los valores en p.u. tienen una interpretación similar a la que se hace sobre los datos cuando se trabajan las cantidades en porcentaje. Por ejemplo, si un voltaje se referencia a su valor nominal, cuando se encuentra en dicho valor tendrá asignado las cantidades 100% o 1 p.u. Cuando se encuentra por debajo del valor nominal entonces asume valores inferiores a 100% o a 1 p.u. y cuando está por encima asume valores superiores a 100% o a 1 p.u. Dado que los niveles máximos y mínimos permitidos están asociados a valores porcentuales, por ejemplo, el voltaje no debe ser superior al 105% ni inferior al 95% del valor nominal, esta relación también puede hacerse en p.u. simplemente dividiendo la cantidad en porcentaje por 100%, en ese ejemplo anterior, entonces el voltaje debe encontrarse entre 0.95 y 1.05 p.u., siendo el valor ideal 1 p.u.

3. Al convertir las cantidades reales a p.u. se elimina el efecto de cambio de tensión producida por los transformadores, en consecuencia, el voltaje en un transformador puede ser 115 KV en primario y 13.2 KV en el secundario y sin embargo puede ser 1.0 p.u en el primario y 1.0 p.u. en el secundario. Esto se denomina comúnmente sistema con voltajes planos.

4. El valor de la impedancia de un transformador vista desde el lado primario resulta igual al valor de la impedancia vista desde el lado secundario si se trabaja en p.u. Esto no ocurre cuando se trabaja con valores reales.

5. Se eliminan los efectos de conexión Y y delta de los transformadores y de las cargas, por lo tanto no es necesario multiplicar o dividir por $\sqrt{3}$ los voltajes o las corrientes, ni tener presente el tipo de conexión utilizado. Esto porque las impedancias en por unidad no dependen del tipo de conexión utilizada en los transformadores en circuitos trifásicos.

6. Si se utilizan cantidades referenciadas en porcentaje debe tenerse cuidado porque al multiplicar dos cantidades en por ciento, el resultado debe dividirse por 100 para que el resultado se encuentre también en por ciento. El producto de dos cantidades en por unidad resulta directamente en por unidad.

7. Generalmente los fabricantes especifican la impedancia de sus equipos en por ciento o en por unidad sobre la base de los valores nominales de los equipos.

8. Los valores de impedancias de equipos del mismo tipo (por ejemplo transformadores) con valores nominales muy diferentes tienen valores de impedancia en ohmios también muy diferentes, sin embargo, si sus impedancias se especifican en p.u. estos valores resultan muy similares. Esta característica es útil para determinar si los valores que se están utilizando en el análisis son apropiados y también sirven para asignar valores típicos de impedancia a un equipo del cual no se tiene información.

El valor en p.u. de una cantidad eléctrica se define como el cociente que resulta de dividir el valor numérico real de dicha cantidad, con su unidad, entre otra cantidad seleccionada arbitrariamente como ***base*** de igual dimensión:

$$X_{p.u.} = \frac{X_{REAL}}{X_{BASE}}$$

Una vez que se dispone de los modelos de los elementos que componen el SEP, este debe representarse interconectado de alguna manera los modelos correspondientes.
Los fabricantes de equipo eléctrico especifican normalmente las características del mismo en forma porcentual o por unidad con respecto a valores nominales, esto es, valores en condiciones de carga u operación normal de diseño.

Debido a la gran diversidad de equipo, surge la necesidad de establecer bases comunes con respecto a las cuales se refieran los parámetros de los circuitos equivalentes, para estar en posibilidad de interconectar los modelos. Esta convención introduce algunas simplificaciones en la representación de los elementos y en la solución computacional.
Un sistema por unidad se especifica expresando la tensión, la corriente, la potencia y la impedancia de un circuito con referencia a un valor base que se elige para cada una de tales magnitudes. El valor por unidad de una magnitud cualquiera se define como la razón de su valor al valor base:

$$Valor\ por\ unidad\ = \frac{Valor\ real}{Valor\ base}$$

El valor base siempre tiene las mismas unidades que el valor real, forzando al valor unitario a ser adimensional. El valor en por ciento es igual a cien veces el valor por unidad. Los métodos de cálculo que utilizan los valores por unidad o por ciento son mucho más sencillos que aquellos que emplean los valores reales en Volts, Ohms, KVA, etc.
Las tensiones, corrientes, potencias e impedancias están relacionadas entre sí, de tal forma que seleccionando dos cantidades base, de entre las cantidades de interés, se pueden encontrar las otras dos. Es común seleccionar el voltaje y la potencia como valores base.

1.4.5 El Diagrama Unifilar o de una Línea.

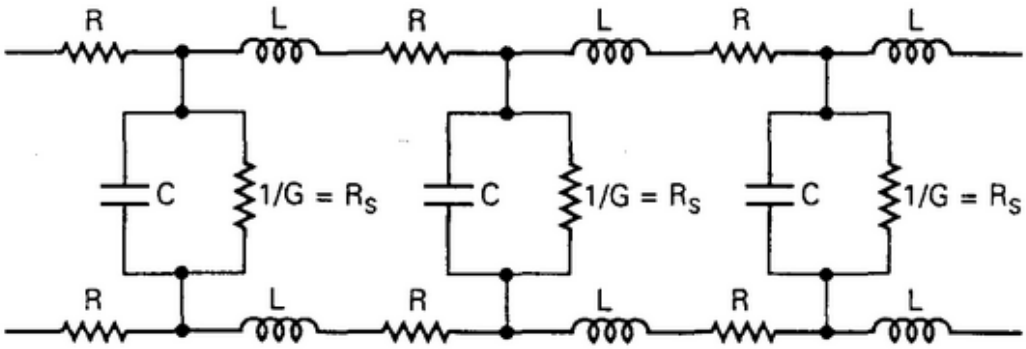

Figura *1.14 Circuito trifásico balanceado*

Siendo:

C = Capacitancia – dos conductores separados por un aislante
R = Resistencia
L = Auto inductancia (Inductancia Propia)
1/G = Resistencia de dispersión eléctrico
R_S = Resistencia de dispersión en derivación

Siempre se resuelve como un circuito equivalente monofásico, o por fase, este diagrama se simplifica al omitir el neutro e indicar las partes que lo componen mediante símbolos estándar en lugar de sus circuitos equivalentes. A este diagrama simplificado de un sistema eléctrico se le llama diagrama unifilar o de una línea.

El propósito de un diagrama unifilar es el de suministrar en forma concisa información significativa acerca del sistema. La importancia de las diferentes piezas de un sistema varía con el problema bajo consideración, y la cantidad de información que se incluye en el diagrama depende del propósito para el que se realiza [51].

En un circuito trifásico balanceado siempre se resuelve como un circuito equivalente monofásico, o por fase, este diagrama se simplifica al omitir el neutro e indicar las partes que lo componen mediante símbolos estándar en lugar de sus circuitos equivalentes. A este diagrama simplificado de un sistema eléctrico se le llama diagrama unifilar o de una línea.

El propósito de un diagrama unifilar es el de suministrar en forma concisa información significativa acerca del sistema. La importancia de las diferentes piezas de un sistema varía con el problema bajo consideración, y la cantidad de información que se incluye en el diagrama depende del propósito para el que se realiza.

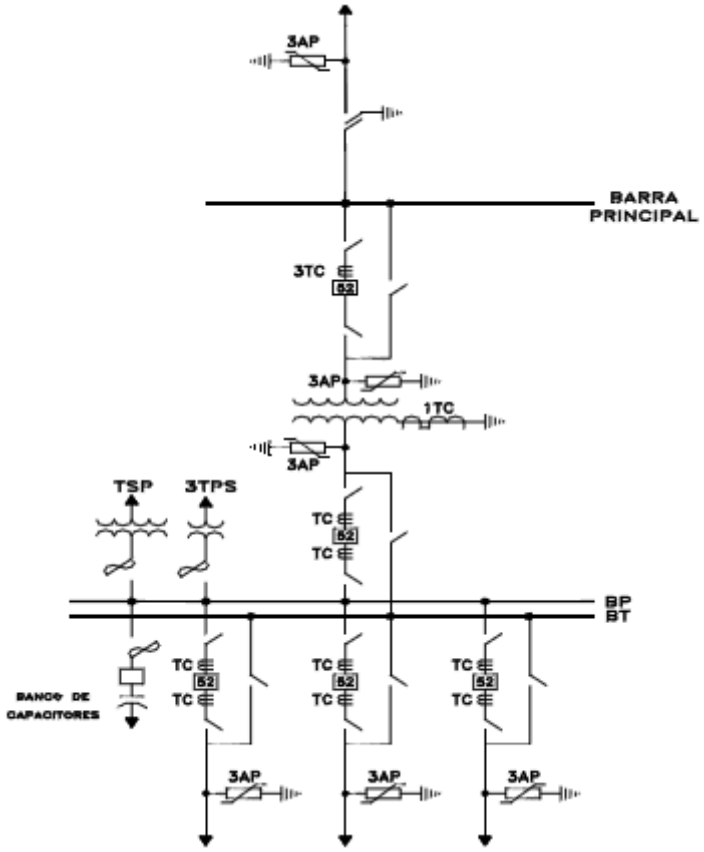

Figura 1.15 ***Diagrama Unifilar de una Subestación Eléctrica [22]***

En el diagrama unifilar se muestran las conexiones entre dispositivos componentes partes de un circuito eléctrico o de un sistema de circuitos, representados mediante símbolos.

Para definir el diseño de la subestación de debe considerar el arreglo de barras y el grado de flexibilidad y confiabilidad requerido en la operación

Existe variación para los arreglos de barras, su selección depende de factores como: tensión del sistema, posición del Sistema Eléctrico en la red, flexibilidad y confiabilidad de operación, continuidad en el suministro y costo de instalación.

Deben cumplir con las especificaciones "diagramas unifilares de arreglos para subestaciones"

Los arreglos a utilizarse son los siguientes:

• Barra Principal

• Barra Principal – Barra de Transparencia

• Anillo

• Arreglo en"H"

CAPITULO II

TÍTULO: FORMACIÓN DE LA MATRIZ DE ADMITANCIA YBUS

2.1 Introducción.

Para el análisis de un sistema eléctrico de potencia en primer lugar debe realizar la formulación de un modelo matemático idóneo. Dicho modelo debe describir las características de los elementos del circuito así como las relaciones asociadas a las interconexiones de cada uno de los componentes.

Para la solución del circuito una de las formas de solución representa una ecuación matricial.

Así mismo, las redes de potencias de los sistemas actuales son de gran dimensión inclusive contienen centenas, tal vez millares de elementos que se interconectan para formar el sistema.

Esto a su vez hace que la solución y la deducción del modelo matemático del sistema de potencia se vuelva más complejo, donde se requiere la ejecución de muchas operaciones matemáticas. Para el hombre se vuelve casi complicado en realizar manualmente dichas operaciones con precisión en tiempo razonable.

Por ello es necesario preparar los datos y adecuarlos en un programa computacional que en este caso se utilizara el programa computacional Matlab.

La matriz que representa la red eléctrica debe presentar dos principales características en su aplicación, es decir en primer lugar debe ser fácil de estructurar con ayuda de un programa computacional y en segundo lugar fácil de modificar.

El análisis nodal se ha convertido a través de los años como la técnica más utilizada para los estudios de los sistemas eléctricos de potencia y puede usarse acertadamente para los análisis de los sistemas de distribución. Lo anterior es el resultado de las ventajas que brinda el almacenamiento y manejo de las matrices dispersas que modelan a las redes eléctricas.

El método de nodos se basa en la aplicación de la Ley de Corriente de Kirchoff o balance de corriente en casa nodo del sistema.

Las variables esenciales son los voltajes nodales y las inyecciones de corriente que aportan los generadores o las corrientes que toman las cargas conectadas en los nodos del sistema.

Metodología.-

Existen varios métodos de montaje que se utiliza en la metodología eléctrica. Entre ellos se ha definido la sobreposición de redes parciales que permiten montar la matriz de red original por unión de matrices de admitancias de subredes parciales, la una conteniendo todas las conexiones no acopladas y la otra, u otras, asociadas a los grupos de mutuas.

2.2 Estudio de la Admitancia.

Se denomina admitancia, de un circuito eléctrico, a la inversa de la impedancia. En forma compleja.

$$\gamma = \frac{1}{Z} \text{ mhos}$$

El módulo de admitancia compleja es el inverso del módulo de la impedancia compleja y su argumento es igual y de signo contrario de la impedancia [32].

$$\text{Si} \qquad Z = Z\angle\varphi = Ze^{j\varphi}$$

La expresión compleja de la admitancia es

$$\gamma = \frac{1}{Z} = \frac{1}{Z}\, e^{j\varphi} = Ye^{-j\varphi}$$

Teniendo en cuenta que

$$e^{j\varphi} = \cos\varphi + \sin\varphi$$

$$e^{-j\varphi} = \cos\varphi - \sin\varphi$$

Resulta que

$$\mathrm{Z} = Z\ \cos\varphi\ + jZ\ \sin\varphi\ = R\ +\ jX$$
$$Y = Y\ \cos\varphi - jY\ \sin\varphi\ = G\ +\ jB$$

Las expresiones

$$\mathrm{G} = \mathrm{Y}\cos\varphi \qquad \text{mhos}$$
$$\mathrm{B} = -Y\ \cos\varphi \quad \text{mhos}$$

Reciben los nombres de conductancia y susceptancia respectivamente

Como se observa, la resistencia y conductancia son magnitudes positivas, mientras que la reactancia y susceptancia son magnitudes de signo contrario. Si la reactancia es debida a una inductancia la susceptancia es debida a la presencia de un condensador y viceversa, fig. 2.1

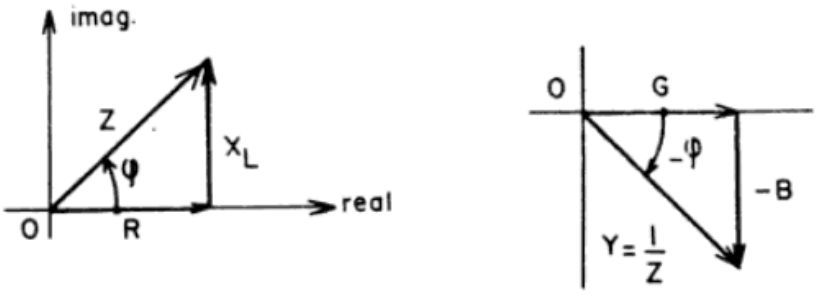

Figura. 2.1 ***Vectores de susceptancia***

Si se conocen las características de un circuito en serie, puede transformare en circuito paralelo equivalente o viceversa, apoyándose en los conceptos de impedancia y admitancia. Para ello, la única condición que deben cumplir los circuitos es que presenten la misma impedancia al generador de alimentación ver fig.2.2

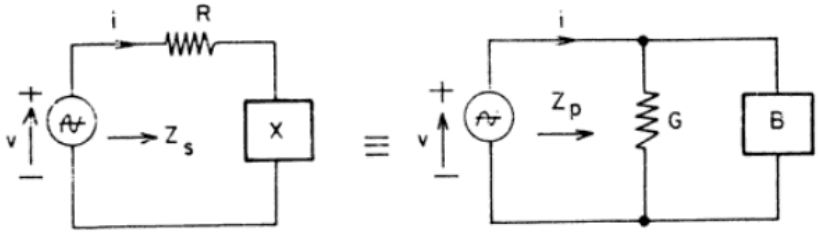

Figura 2.2 ***Equivalencia serie o paralelo***

Al ser $$Z_P = Z_S = R + jX = \frac{1}{G + j X}$$

Se tiene que

$$Y_P = \frac{1}{Z_S} = G + jB = \frac{1}{R + jX}$$

Luego

$$G + jB = \frac{R - jX}{R^2 + X^2} = \frac{R}{R^2 + X^2} - j\frac{X}{R^2 + X^2}$$

Por tanto:

$$G = \frac{R}{R^2 + X^2}$$

$$B = -\frac{X}{R^2 + X^2}$$

$$(ohmios)^{-1} = mhos$$

Inversamente

$$Z_S = \frac{1}{Y_P} = R + jX = \frac{1}{G + jB}$$

De donde

$$R + jX = \frac{G - jB}{G^2 + B^2} = \frac{G}{G^2 + B^2} - j\frac{B}{G^2 + B^2}$$

Por tanto

$$R = \frac{G}{G^2 + B^2}$$

$$X = \frac{-B}{G^2 + B^2}$$

2.3 Formación de la matriz de Admitancias, aplicación de las de Kirchhof

La matriz de admitancia también llamada Y_{barra} o Y_{nodo} , los elementos de Y_{ij} serán: *i* y *j* la fila y la columna correspondientes de la matriz. El método de la matriz de impedancias permite determinar la corriente de falla de todos los elementos y los voltajes existentes en cada bus después de que la falla ocurrió.

El método consiste en lo siguiente:

Primeramente se debe tener el diagrama unifilar del sistema a analizar ya con las reactancias referidas al valor base elegido. En este diagrama se deben poner los valores de las impedancias (reactancias) como valores de admitancias (susceptancias). Con estos valores ya podemos formar la matriz de admitancias de la siguiente forma:

1.- Admitancias propias: Los elementos que formarán la diagonal principal, se calculan sumando todas las admitancias de los elementos unidos al bus a analizar.

2.- Admitancias mutuas: serán las admitancias equivalentes de los elementos existentes entre un par de buses considerados. La matriz de admitancias de bus siempre es simétrica, su orden lo determina el número de buses del sistema, todas las admitancias propias tienen signo negativo y todas las admitancias mutuas signo positivo.

La imagen de la Figura 2.3 muestra un sistema eléctrico con tres buses, veremos el procedimiento para formar la matriz de admitancias.

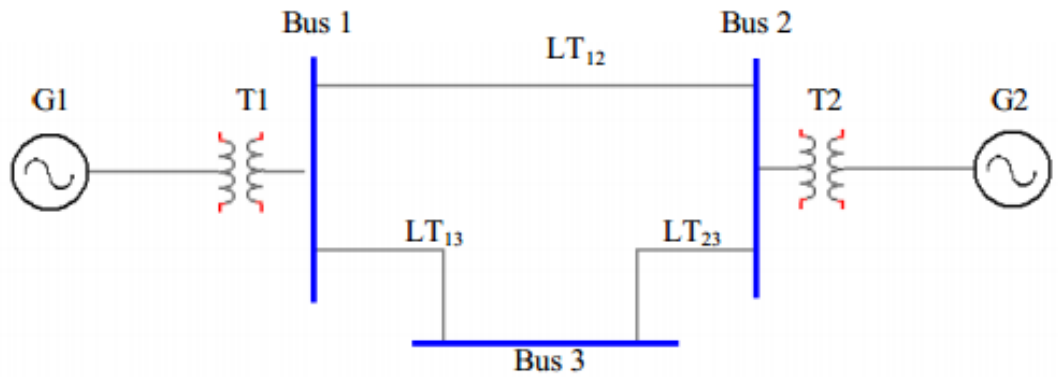

Figura 2.3 ***Sistema de potencia compuesto por 3 buses.***

El diagrama de impedancias del sistema anterior se muestra en la figura 2.4

Como lo dicta el procedimiento, cambiamos los valores de las impedancias en valores de admitancias, esto es:

$$Y_a = \frac{1}{X_a} \quad Y_b = \frac{1}{X_b} \quad Y_c = \frac{1}{X_c} \quad Y_d = \frac{1}{X_d} \quad Y_e = \frac{1}{X_e}$$

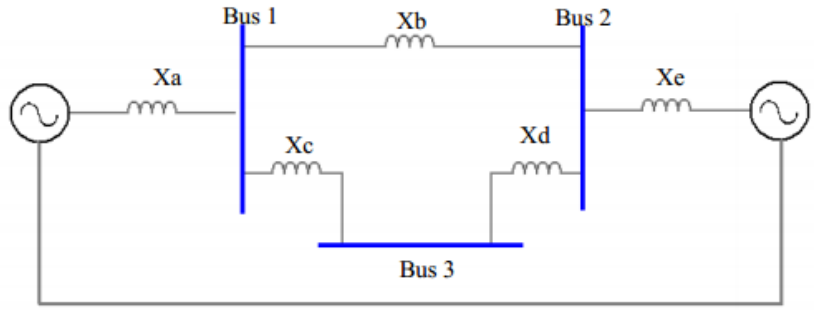

Figura 2.4 ***Sistema representado por sus impedancias***

Como lo dicta el procedimiento, cambiamos los valores de las impedancias en valores de admitancias, esto es:

$$Y_a = \frac{1}{X_a} \quad Y_b = \frac{1}{X_b} \quad Y_c = \frac{1}{X_c} \quad Y_d = \frac{1}{X_d} \quad Y_e = \frac{1}{X_e}$$

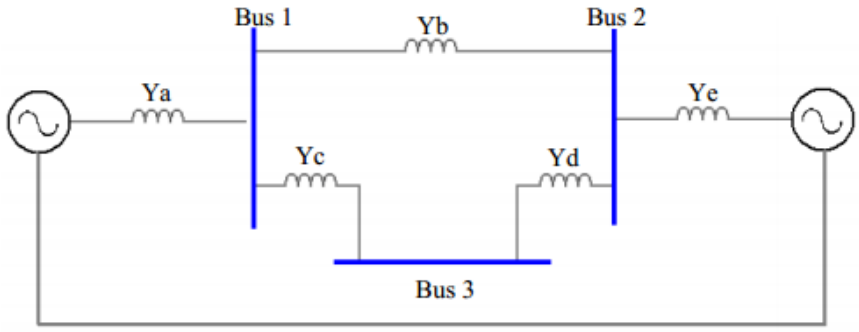

Fig.2.5 Sistema eléctrico representado por sus admitancias.

Y según el análisis de nodos la matriz de admitancias queda de la siguiente forma:

$$\boldsymbol{Y_{Barra}} = \begin{bmatrix} Y_{11} & Y_{12} & Y_{13} \\ Y_{21} & Y_{22} & Y_{23} \\ Y_{31} & Y_{32} & Y_{33} \end{bmatrix}$$

Sustituyendo los valores correspondientes a cada admitancia de bus obtenemos:

$$\boldsymbol{Y_{11}} = Y_a + Y_b + Y_c \qquad \boldsymbol{Y_{22}} = Y_b + Y_d + Y_e$$

$$\boldsymbol{Y_{12}} = Y_{21} = -Y_b \qquad \boldsymbol{Y_{23}} = Y_{32} = -Y_d$$

$$\boldsymbol{Y_{13}} = Y_{31} = -Y_c \qquad \boldsymbol{Y_{33}} = Y_c + Y_d$$

La matriz queda de la siguiente forma:

$$\boldsymbol{Y_{Barra}} = \begin{bmatrix} (Y_a + Y_b + Y_c) & -Y_b & -Y_c \\ -Y_b & (Y_b + Y_d + Y_e) & -Y_d \\ -Y_c & -Y_d & (Y_c + Y_d) \end{bmatrix}$$

Aplicación de las leyes de Kirchof

La ley de los nudos dice que "en todo nudo eléctrico, la suma vectorial de las corrientes que a el se acercan es igual a la suma vectorial de las corrientes que se alejan de el" [43].

La ley de mallas dice que "en todo nudo eléctrico, la suma vectorial de las fuerzas electromotrices aplicadas es igual a la suma vectorial de las caídas de tensión que en ella se producen ". Se pueden obtener las funciones características de una red lineal activa, tales como la función de transferencia, impedancia de entrada, etc., utilizando los métodos clásicos de mallas y nodos (leyes de voltaje y corriente de Kirchhoff). Sin embargo, este procedimiento puede conducir a relaciones muy complicadas y laboriosas que hacen difícil obtener un resultado de forma sistemática. Afortunadamente, hay modos muy directos en los cuales se puede derivar las funciones de la red desde el diagrama circuital, sin necesidad de desarrollar los pasos intermedios de las ecuaciones de la red. Uno de los métodos de análisis más útiles, emplea la matriz indefinida de admitancias. Este método proporciona la matriz de los parámetros de admitancia para la caracterización de una red multiterminal por simple inspección de la misma. La expansión de la matriz por métodos convencionales, proporciona la función deseada de la red.

En este punto cabe traer a colación el método de Zbus. La Zbus es una matriz de impedancias. Como nuestras impedancias son igual a la resistencia, entonces nuestra matriz contendrá todas las resistencias del circuito. Pero para que este método funcione hay que construir primero una matriz de admitancias, lo que llamamos Ybus.

Utilizando el post de la Ley de Las Corrientes de Kirchhoff.

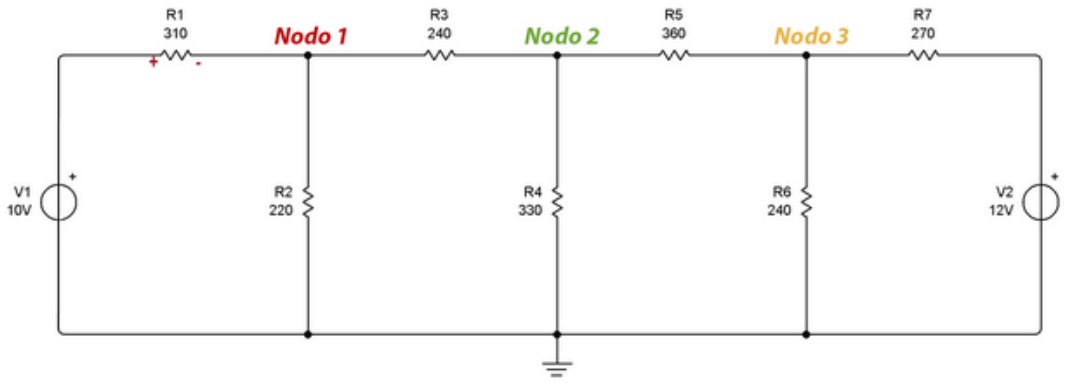

Figura ***2.6 Nodo 2 y Nodo 2***

El patrón para construir la Ybus es el siguiente:

$$\boldsymbol{Y_{Bus}} = \begin{bmatrix} Y_{nodo\ 1} & -Y_{1-2} & -Y_{1-3} \\ -Y_{2-1} & Y_{nodo\ 2} & -Y_{2-3} \\ -Y_{3-1} & -Y_{3-2} & Y_{nodo\ 3} \end{bmatrix}$$

Ahora empezamos a llenar nuestra Ybus.

En la primera fila se coloca todas las admitancias relacionadas con el nodo 1. En el primer índice se coloca todas las admitancias que van al nodo 1. En el segundo índice (1-2) todas las admitancias entre el nodo 1 y el nodo 2 con signo invertido. En el tercer índice todas las admitancias entre el nodo 1 y el nodo 3, con signo invertido.

Veamos las admitancias que tocan el nodo 1:

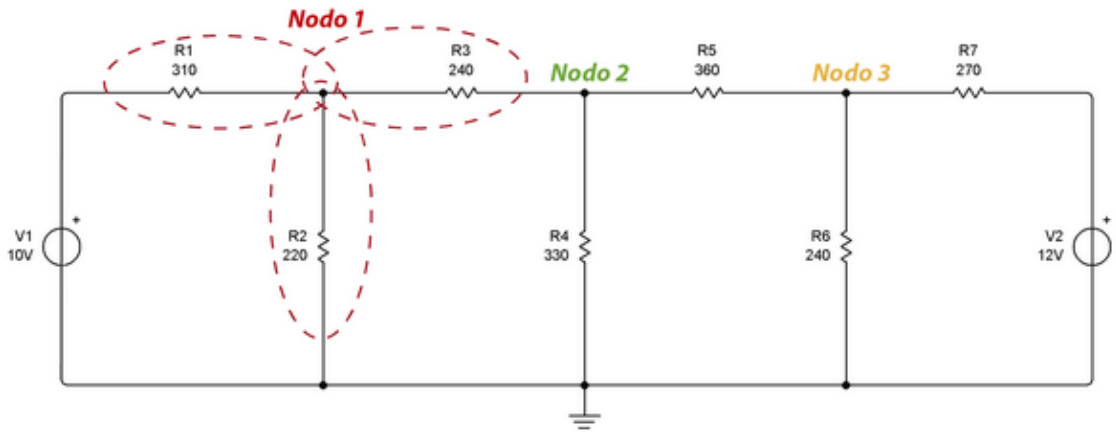

Figura 2.7 ***Desarrollo en nodo 1***

Vemos que hay 3 resistencias que tocan el nodo 1. Pero no necesitamos las resistencias sino las admitancias, así que las invertimos y las sumamos.

$$Y_{Bus} = \begin{pmatrix} \frac{1}{310} + \frac{1}{220} + \frac{1}{240} & -Y_{1-2} & -Y_{1-3} \\ -Y_{2-1} & Y_{nodo\ 2} & -Y_{2-3} \\ -Y_{3-1} & -Y_{3-2} & Y_{nodo\ 3} \end{pmatrix}$$

Ahora vemos las admitancias entre el nodo 1 y el nodo 2.

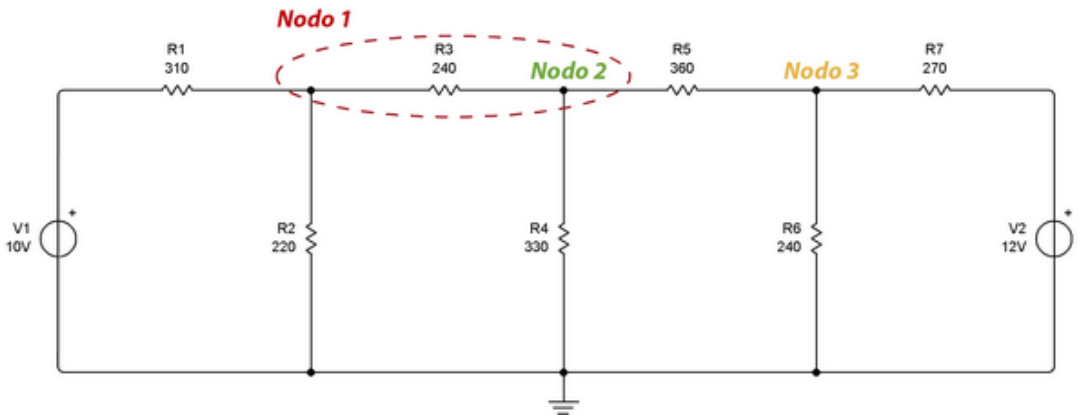

Figura ***2.8 Admitancias en nodo 1***

Solo hay una resistencia entre el nodo 1 y el nodo 2, así que solo colocamos esa con signo invertido. Como entre el nodo 1 y el nodo 3 no hay conexión directa, se considera la impedancia 1-3 como cero.

$$Y_{BUS} = \begin{pmatrix} \frac{1}{310} + \frac{1}{220} + \frac{1}{240} & \frac{-1}{240} & 0 \\ -Y_{2-1} & Y_{nodo\ 2} & -Y_{2-3} \\ -Y_{3-1} & -Y_{3-2} & Y_{nodo\ 3} \end{pmatrix}$$

Ahora hacemos lo mismo para todos y cada uno de los nodos y obtenemos nuestra Ybus.

$$Y_{BUS} = \begin{pmatrix} \frac{1}{310} + \frac{1}{220} + \frac{1}{240} & \frac{-1}{240} & 0 \\ -\frac{1}{240} & \frac{1}{240} + \frac{1}{330} + \frac{1}{360} & -\frac{1}{360} \\ 0 & -\frac{1}{360} & \frac{1}{360} + \frac{1}{270} + \frac{1}{240} \end{pmatrix}$$

Como el inverso de la admitancia es la impedancia, si invertimos la Ybus se obtiene la Zbus [54].

Al final nuestra Zbus es la siguiente:

$$Z_{BUS} = \begin{pmatrix} 99.39 & 44.77 & 11.68 \\ 44.77 & 128.27 & 33.46 \\ 11.67 & 33.46 & 102.64 \end{pmatrix}$$

Ahora por medio de la Ley de Ohm podemos encontrar los voltajes en los nodos.

Como sabemos que el Voltaje es el producto de la Corriente por la Resistencia, podemos deducir que para este caso el Voltaje también es igual a la corriente por la Impedancia.

Dicho esto tenemos: $V = IZ$

$$Z_{BUS} = \begin{pmatrix} 99.39 & 44.77 & 11.68 \\ 44.77 & 128.27 & 33.46 \\ 11.67 & 33.46 & 102.64 \end{pmatrix}$$

$$V = \begin{pmatrix} V_{nodo\ 1} \\ V_{nodo\ 2} \\ V_{nodo\ 3} \end{pmatrix}$$

$$I = \begin{pmatrix} I_{nodo\ 1} \\ I_{nodo\ 2} \\ I_{nodo\ 3} \end{pmatrix}$$

$$\begin{pmatrix} V_{nodo\ 1} \\ V_{nodo\ 2} \\ V_{nodo\ 3} \end{pmatrix} = \begin{pmatrix} 99.39 & 44.77 & 11.68 \\ 44.77 & 128.27 & 33.46 \\ 11.67 & 33.46 & 102.64 \end{pmatrix} \begin{pmatrix} I_{nodo\ 1} \\ I_{nodo\ 2} \\ I_{nodo\ 3} \end{pmatrix}$$

En esta igualdad tenemos todo excepto la corriente en los nodos. Esta corriente es fácil de obtener si dividimos el voltaje de las fuentes entre las resistencias que tocan los nodos y los unen con las fuentes.

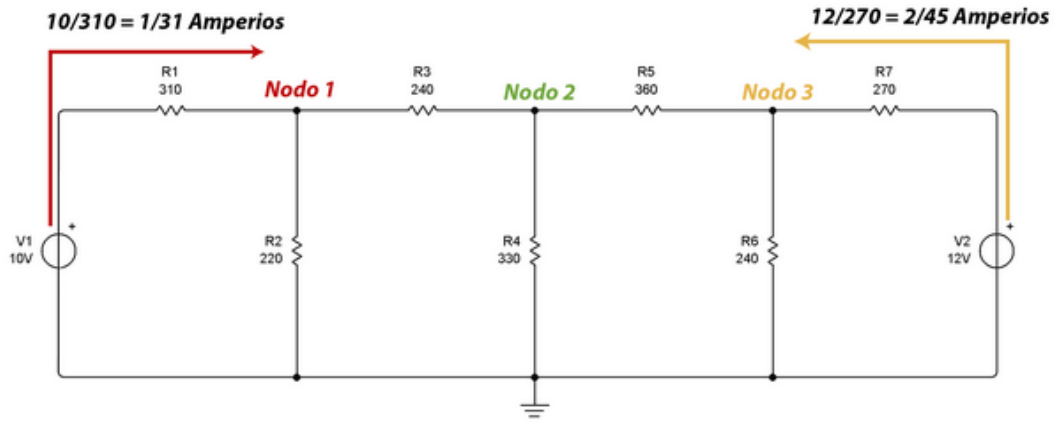

Figura 2.9 ***Corrientes de fuentes***

Al nodo 2 no hay ninguna corriente entrando por la acción directa de una fuente de voltaje o de corriente, por lo que se asume que es cero. Ahora construimos nuestra matriz de corrientes.

$$\begin{pmatrix} V_{nodo\ 1} \\ V_{nodo\ 2} \\ V_{nodo\ 3} \end{pmatrix} = \begin{pmatrix} 99.39 & 44.77 & 11.68 \\ 44.77 & 128.27 & 33.46 \\ 11.67 & 33.46 & 102.64 \end{pmatrix} \begin{pmatrix} \frac{1}{31} \\ 0 \\ \frac{2}{45} \end{pmatrix}$$

Ahora podemos ejecutar esta operación, utilizando Matlab, declaro las matrices de voltaje y de corriente.

Figura 2.10 ***Resultado de voltaje y corriente desarrollando Matlab***

Ahora se declara la Matriz Ybus.

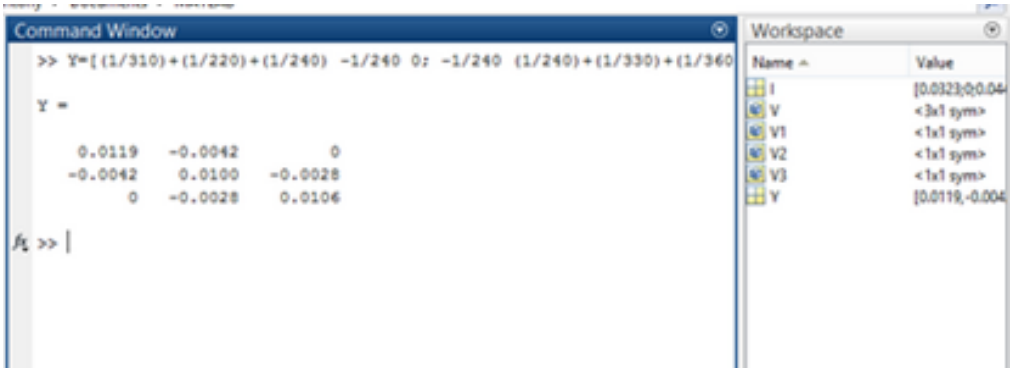

Figura 11 ***Determinación de Matriz YBus desarrollando Matlab***

Ahora se ejecuta el comando I*inv(Y)==V y vemos el resultado:

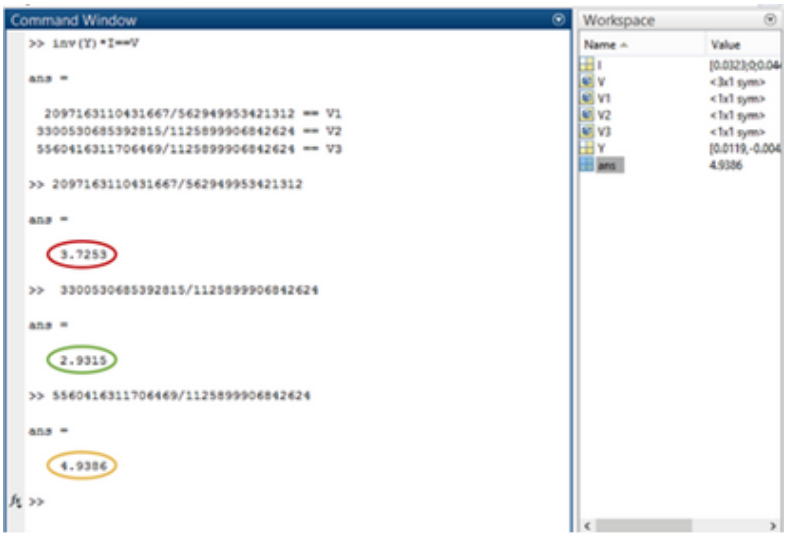

Figura 2.12 ***Resultado numérico aplicando Matlab***

Comprobamos los resultados en el simulador:

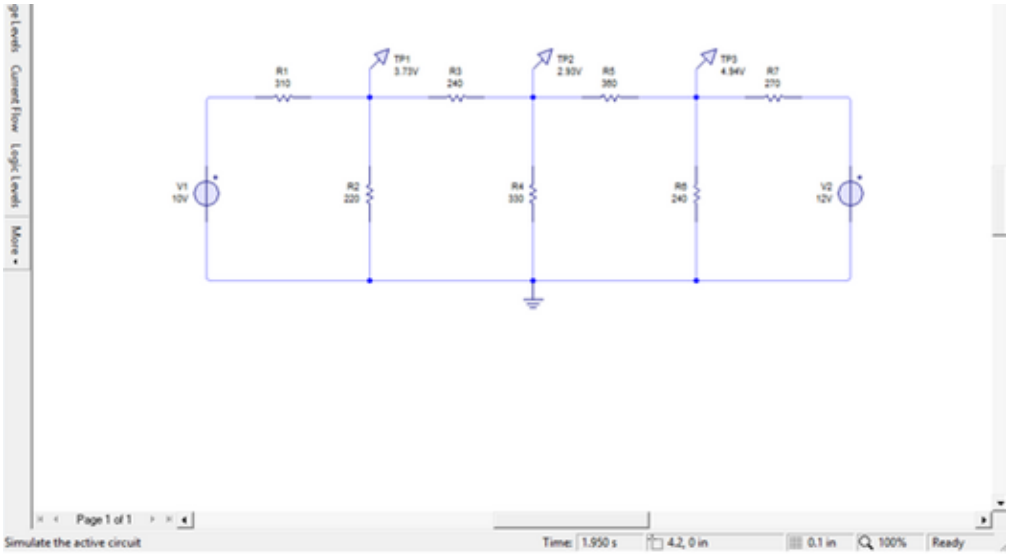

Figura 2.13 ***Simulando circuito aplicando Matlab***

El método de Zbus es el más sencillo de todos ya que no se requiere de mucho análisis, solo hay que mirar los nodos y las resistencias que llegan a cada uno. Se transforman a admitancias y se suman. Se consigue la corriente que entra por el efecto de fuentes de voltaje a cada nodo.
Se establecen las matrices y con una buena herramienta de cálculo se pueden obtener los voltajes en los nodos de manera fácil y sencilla.

2.3.1 Formación de la Matriz de Admitancia por las aplicaciones de matrices de Transformaciones e inspección de la Red.

En el análisis de sistemas eléctricos es necesario disponer de todos los datos para llevar a cabo una gran cantidad de estudios que permiten determinar sus condiciones de operación tanto en estado estacionario como en estado transitorio [44].

Para ello es importante conocer las matrices de impedancias y admitancias de la red, debido a que así es posible, mediante estudios de flujos de potencia calcular los voltajes de cada nodo de la red, así como la potencia real y reactiva que circula a través de los sistemas de transmisión, o al llevar a cabo un estudio de corto circuito, para determinar la corriente en puntos determinados con la finalidad de coordinar adecuadamente los dispositivos de protección y aislar los puntos fallados.

En el estudio de sistemas eléctricos, es indispensable conocer las impedancias de secuencia de cada uno de los componentes que forman parte de él, esto es; transformadores, generadores, grandes motores, así como la impedancia de los sistemas de transmisión y distribución. Todos estos datos son necesarios debido a que cada uno de ellos forma parte de las matrices de impedancias y admitancias de la red y realizar así una gran cantidad de estudios en la red eléctrica. Se presentan diferentes formas de calcular la matriz de impedancias y la matriz de admitancias de una red eléctrica [41].

La matriz de admitancias del sistema eléctrico se lo nombra también como Ybus, Ybarra, o Ynodo, lo importante es tener pleno conocimiento de cómo formarla y en que estudios es utilizada [21]. Por el momento se le denominara simplemente la matriz de *ij*, admitancias Y, y los elementos de la misma como y siendo i, y j la fila y columna correspondiente de la matriz.

Tenemos una red de tres nodos más el de referencia como la ilustrada en la Figura 2.14 en la que se han etiquetado todos los puntos nodales, se tiene la matriz de admitancias que a continuación se describe:

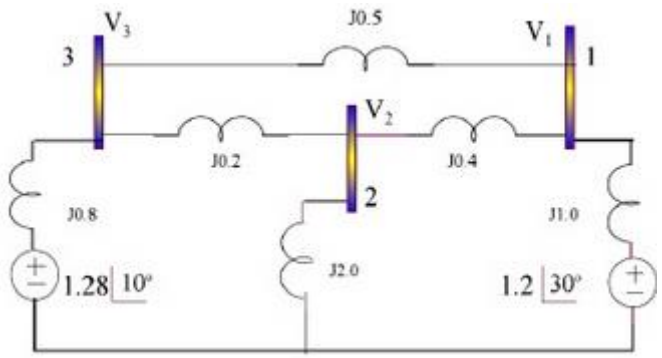

Figura ***2.14 Red eléctrica de tres nodos***

Transformando las fuentes de voltaje en serie con sus impedancias a fuentes de corriente en paralelo con sus respectivas impedancias, y se analiza cada nodo por separado, se tiene para el nodo uno la representación mostrada en la Figura 2.15

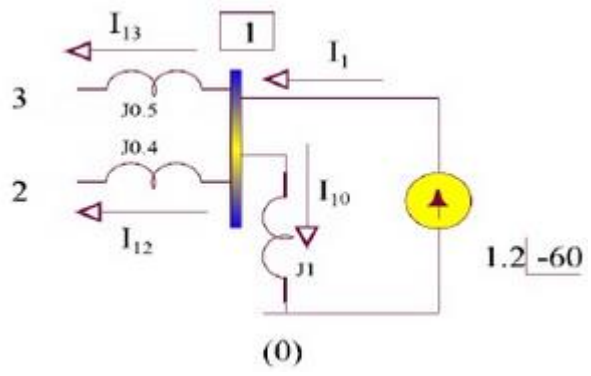

Figura 2.15 ***Nodo 1 se muestran corrientes que entran y salen***

En la Figura 2.15, se han dibujado las corrientes que entran y salen del nodo, haciendo uso de la notación con doble subíndice para indicar que el primero tiene un potencial mayor que el segundo y que la corriente se supone que fluye en la dirección mostrada del nodo uno con los demás nodos con los cuales tiene conexión. Así, la aplicación de la ley de corrientes de Kirchhoff aplicada al nodo uno permite establecer la ecuación siguiente

$$I_1 = I_{13} + I_{12} + I_{10}$$

Recordando que el fasor corriente I se expresa también como la diferencia de potencial Vsobre la impedancia entre el nodo i y el nodo j, la ecuación anterior se escribe de la forma siguiente

$$I_1 = \frac{V_1 - V_3}{Z_{13}} + \frac{V_1 - V_2}{Z_{12}} + \frac{V_1}{Z_{10}}$$

Realizando factorización se tiene

$$I_1 = V_1\left[\frac{1}{Z_{12}} + \frac{1}{Z_{13}} + \frac{1}{Z_{10}}\right] - \frac{1}{Z_{12}}V_2 - \frac{1}{Z_{13}}V_3$$

Aplicando la ley de corrientes al nodo dos, como se muestra en la Figura 2.16

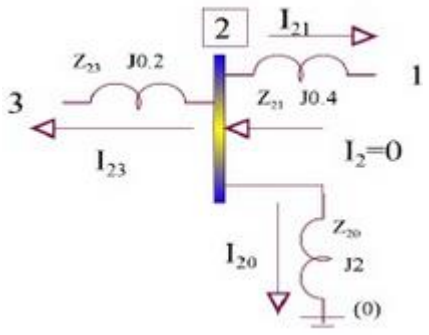

Figura 2.16 ***Nodo 2 se muestran corrientes que entran y salen***

Aplicamos mismo proceso y luego factorizando se tiene

$$I_2 = -\frac{1}{Z_{21}}V_1 + V_2\left[\frac{1}{Z_{20}} + \frac{1}{Z_{12}} + \frac{1}{Z_{23}}\right] - \frac{1}{Z_{23}}V_3$$

Para el nodo tres, después de transformar la fuente de voltaje en serie con la impedancia de j0.8, a una fuente de corriente en paralelo con la misma impedancia, se tiene la representación mostrada en la Figura 2.17

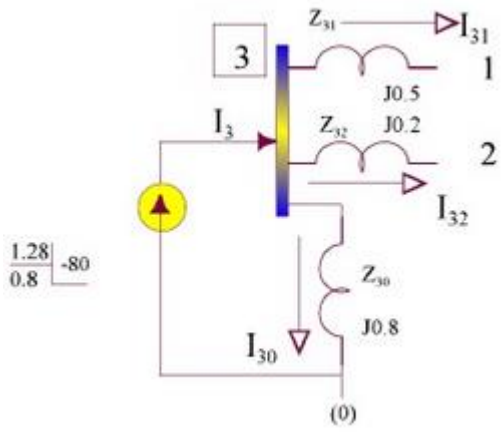

Figura 2.17 Nodo 3 se muestran corrientes que entran y salen

Aplicando las leyes de Kirchhoff, sé tienen las siguientes ecuaciones

$$I_3 = I_{30} + I_{31} + I_{32}$$

Para términos de voltaje e impedancia se tiene

$$I_3 = \frac{V_3}{Z_{30}} + \frac{V_3 - V_1}{Z_{31}} + \frac{V_3 - V_2}{Z_{32}}$$

Agrupando en términos semejantes se tiene:

$$I_3 = -\frac{1}{Z_{31}}V_1 - \frac{1}{Z_{32}}V_2 + V_3\left[\frac{1}{Z_{30}} + \frac{1}{Z_{31}} + \frac{1}{Z_{32}}\right]$$

Las variables a determinar son los voltajes de los nodos 1,2, y 3

$$\begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix} = \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix} \begin{pmatrix} \frac{1}{Z_{10}} + \frac{1}{Z_{12}} + \frac{1}{Z_{13}} & \frac{-1}{Z_{12}} & -\frac{1}{Z_{13}} \\ -\frac{1}{Z_{21}} & \frac{1}{Z_{20}} + \frac{1}{Z_{21}} + \frac{1}{Z_{23}} & -\frac{1}{Z_{23}} \\ -\frac{1}{Z_{31}} & -\frac{1}{Z_{32}} & \frac{1}{Z_{30}} + \frac{1}{Z_{31}} + \frac{1}{Z_{32}} \end{pmatrix}$$

En forma compacta se acostumbra a escribir la ecuación anterior de la forma:

En dónde:

[Y] Es la matriz de admitancias.

V es el vector de voltajes nodales.

I es el vector de corrientes aplicadas a cada nodo

Los sistemas eléctricos reales normalmente están formados por un considerable número de nodos, por lo que no es cómodo establecer para cada uno la ley de corrientes de Kirchhoff y encontrar una relación, en su lugar se acostumbra a tener la información de la red como se muestra en la tabla 2.1

N°	Nodo P	Nodo Q	Impedancia	Admitancia Paralelo
1	1	0	$0 + j1.0(Z_{11})$	j0.0
2	1	2	$0 + j0.4(Z_{12})$	j0.0
3	1	3	$0 + j0.5(Z_{13})$	j0.0
4	2	0	$0 + j2.0(Z_{31})$	j0.0
5	2	3	$0 + j0.2(Z_{32})$	j0.0
6	3	0	$0 + j0.8(Z_{30})$	j0.0

Tabla 2.1 ***Información de interconexión de la red.***

En la tabla 2.1 se muestra toda la información necesaria de la red para la formación sistemática de admitancias, la cual es aplicable independientemente del tamaño del sistema. El método deformación de la matriz de admitancias se denomina así, debido a que únicamente es necesario observar detenidamente la red o los datos para determinar el valor de los elementos de [Y] dela ecuación (13)

El método es aplicable de la forma siguiente: Analizando la tabla 1, sin considerar el nodo de referencia (0), la admitancia propia del nodo 1está formada por los elementos 1, 2, y 3. De tal manera que:

$$Y_{11} = \frac{1}{j1} + \frac{1}{j0.4} + \frac{1}{j0.5}$$

De igual manera, para el nodo dos, la admitancia y está formada por los elementos 2, 4, y 5, y es igual a:

$$Y_{22} = \frac{1}{j0.4} + \frac{1}{j2} + \frac{1}{j0.2}$$

Finalmente, para el nodo tres, su admitancia está formada por los elementos 3, 5, y 6, y es igual a:

$$Y_{33} = \frac{1}{j0.5} + \frac{1}{j0.2} + \frac{1}{j0.8}$$

Los elementos y de la matriz de admitancias se obtienen por la observación de los datos en la tabla 1, basta con analizar las columnas P y Q sin considerar el elemento cuando Q=0.

Así, para el elemento dos en el que P=1 y Q=2, se tiene que:

$$Y_{P2} = Y_{12} = \frac{1}{Z_{12}} = -\frac{1}{j0.4}$$

En el elemento tres, se tiene P=1, y Q=3, por lo que:

$$Y_{PQ} = Y_{13} = \frac{1}{Z_{13}} = -\frac{1}{j0.5}$$

En el elemento cinco, se tiene P=2, y Q=3, por lo que:

$$Y_{PQ} = Y_{23} = -\frac{1}{Z_{23}} = -\frac{1}{j0.2}$$

El signo negativo en las admitancias es debido a que la corriente entre el nodo i y el nodo j, queda determinada por la diferencia del voltaje del nodo i y el nodo j, de donde aparece el término -Vj/Zij. La matriz de admitancias pertenece a una red bilateral lineal en donde se cumple que;

$$Y_{21} = Y_{12}\ ,\ Y_{31} = Y_{13}\ ,Y_{32} = Y_{23}$$

Una forma sistemática y rápida para encontrar la matriz de admitancias por inspección a partir de los datos de la tabla 1 es la siguiente:
Para los elementos de la diagonal principal, la admitancia propia es igual a:

$$Y_{ii} = \sum_{k=1}^{E_1} \frac{1}{Z_k} \qquad i = 1\ , \qquad n$$

En dónde:

E_1 Es un número de elementos conectados al nodo i,
Z_k Es la impedancia conectada al nodo i,
Y_{ii} y es la admitancia propia del nodo i

En palabras “La admitancia propia de cada nodo i de la matriz [Y], es igual a la suma de los inversos de las impedancias de los elementos conectados a ese nodo” [23].
Las admitancias colocadas fuera de la diagonal principal de la matriz de admitancias se obtienen a partir de la relación siguiente:

$$Y_{PQ} = Y_{PQ} = -\frac{1}{Z_{PQ}}$$

En dónde:

Los nodos P y Q deben ser diferentes al nodo de referencia,
P es el índice del nodo de inicio,
Q es el nodo final.
Y_{ij} Es la admitancia mutua entre el nodo i y el nodo j

De esta forma, la ecuación establece que:" La admitancia mutua entre el nodo i y el nodo j, es igual al negativo del inverso de la impedancia entre eso nodos

.

2.4 Singulares, Ejemplos

En el sistema de la figura se pretende hacer un flujo de carga linealizado

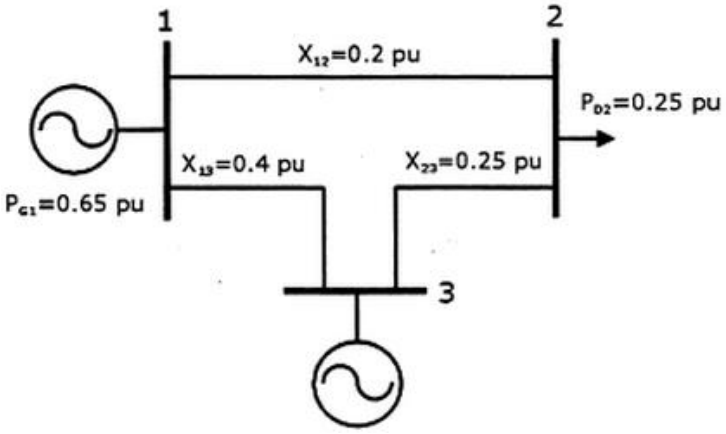

Figura 2.18 ***Circuito eléctrico ejercicio 1***

En primer lugar se identifica el nudo de referencia, que resulta ser el 3
Las matrices de la ecuación [P] = [Bx] . [ð] son por tanto:

$$[\boldsymbol{P}] = \begin{bmatrix} \boldsymbol{P_1} \\ \boldsymbol{P_2} \\ \boldsymbol{0} \end{bmatrix}, [\boldsymbol{ð}] = \begin{bmatrix} \boldsymbol{ð_1} \\ \boldsymbol{ð_2} \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{y}\ [\boldsymbol{B_X}] = \begin{bmatrix} \boldsymbol{\frac{1}{x_{12}} + \frac{1}{x_{13}}} & \boldsymbol{-\frac{1}{x_{13}}} & \boldsymbol{0} \\ \boldsymbol{-\frac{1}{x_{13}}} & \boldsymbol{\left(\frac{1}{x_{21}} + \frac{1}{x_{23}}\right)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}$$

Sustituyendo valores: $\begin{bmatrix} 0.65_1 \\ -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 7.5 & -5 & 0 \\ -5 & 9 & 0 \\ 0 & 0 & 0 \end{bmatrix} . \begin{bmatrix} \delta_1 \\ \delta_2 \\ 0 \end{bmatrix}$

Resolviendo $\begin{bmatrix} \delta_1 \\ \delta_2 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.02 \\ -0.1 \\ 0 \end{bmatrix} (rad)$

Los flujos por las líneas se calculan mediante la expresión:

$$P_{ij} \cong \frac{1}{x_{ij}} \left(\delta_i - \delta_j\right)$$

$$P_{12} \cong \frac{1}{x_{12}} (\delta_1 - \delta_2) = 0.6\, pu.$$

$$P_{13} \cong \frac{1}{x_{13}} (\delta_1 - \delta_3) = 0.05\, pu.$$

$$P_{23} \cong \frac{1}{x_{23}} (\delta_2 - \delta_3) = -0.4\, pu.$$

Ahora colocando en la figura:

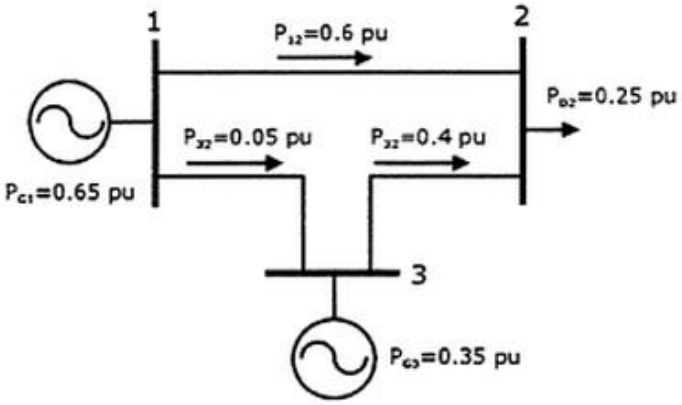

Figura 2.19 Resumiendo valores del ejercicio 1

Observaciones:

Se trata de una aproximación al flujo de potencias activas

No permite calcular los flujos de reactivas

No permite calcular perdidas en las líneas

Supone un perfil plano de tensiones

Problema N°2

En la red de la figura y suponiendo una potencia base de 100 MVA.

a) **Determine mediante los algoritmos G-S las tensiones en los nudos 2 y 3.**
b) **Potencia activa y reactiva en el nudo de referencia**
c) **Flujos y pérdidas en las líneas.**

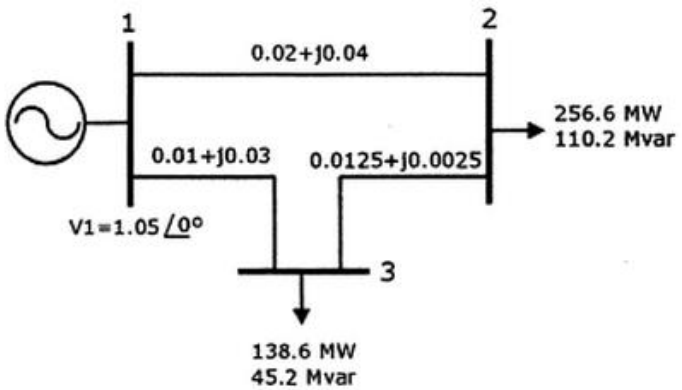

Figura 2.20 Circuito eléctrico ejercicio 2

a.- Convirtiendo las impedancias de líneas en admitancias:

$$y_{12} = 10 - j20; \quad y_{13} = 10 - j30; \qquad y_{23} = 16 - j32$$

La potencia compleja en valores por unidad en los nudos de carga resulta ser:

$$S_2 = -\frac{256.6 + J110.2}{100} = -2.566 - j1.102\ pu$$
$$S_3 = -\frac{138.6 + J45.2}{100} = -1.386 - j0.452\ pu$$

Comenzando con una estimación inicial de tensiones en los nudos 2 y 3 de 1.0 p.u.:

$$V_2^{(1)} = \frac{\frac{P_2 - jQ_2}{V_2^{(0)}} + y_{12}V_1 + y_{23}V_3^{(0)}}{y_{12} + y_{23}} = 0.9825 - j0.031$$

$$V_3^{(1)} = \frac{\frac{P_3 - jQ_3}{V_3^{(0)}} + y_{13}V_1 + y_{23}V_2^{(1)}}{y_{12} + y_{23}} = 1.011 - j0.0353$$

En la segunda iteración:

$$V_2^{(2)} = \frac{\frac{P_2 - jQ_2}{V_2^{(1)}} + y_{12}V_1 + y_{23}V_3^{(1)}}{y_{12} + y_{23}} = 0.9816 - j0.052$$

$$V_3^{(2)} = \frac{\frac{P_3 - jQ_3}{V_3^{(0)}} + y_{13}V_1 + y_{23}V_2^{(2)}}{y_{12} + y_{23}} = 1.0008 - j0.0459$$

En las siguientes iteraciones:

$$V_2^{(2)} = \frac{\frac{P_2 - jQ_2}{V_2^{(1)}} + y_{12}V_1 + y_{23}V_3^{(1)}}{y_{12} + y_{23}} = 0.9816 - j0.052$$

$$V_3^{(2)} = \frac{\frac{P_3 - jQ_3}{V_3^{(1)}} + y_{13}V_1 + y_{23}V_2^{(2)}}{y_{12} + y_{23}} = 1.0008 - j0.0459$$

En las siguientes interacciones:

$V_2^{(3)} = 0.9808 - j0.0578$ $\quad$ $V_3^{(3)} = 1.004 - j0.0488$

$V_2^{(4)} = 0.9803 - j0.0594$ $\quad$ $V_3^{(4)} = 1.002 - j0.0497$

$V_2^{(5)} = 0.9801 - j0.0598$ $\quad$ $V_3^{(5)} = 1.001 - j0.0499$

$V_2^{(6)} = 0.9801 - j0.0599$ $\quad$ $V_3^{(6)} = 1.000 - j0.0500$

$V_2^{(7)} = 0.9800 - j0.0600$ $\quad$ $V_3^{(7)} = 1.000 - j0.0500$

b) Conocidas todas las tensiones, la potencia completa en el nudo 1 será:

$$P_1 - jQ_1 = V_1^*[V_1(y_{12} + y_{13}) - y_{12}V_2 - y_{13}V_3] = 4.095 - j1.890$$

c) Previamente es necesario conocer la intensidad circulante en cada línea:

$$I_{12} = y_{12}(V_1 - V_2) = 1.9 - j0.8 \qquad I_{21} = -I_{12} = -1.9 + j0.8$$

$$I_{13} = y_{13}(V_1 - V_3) = 2.0 - j1.0 \qquad I_{31} = -I_{13} = -2.0 + j1.0$$

$$I_{23} = y_{23}(V_2 - V_3) = -0.64 + j0.48 \qquad I_{23} = -I_{32} = 0.64 - j0.48$$

Los flujos en cada línea son entonces:

$$S_{12} = V_1 I_{12}^* = 199.5\ MW + j84.0\ MVAR$$

$$S_{21} = V_2 I_{21}^* = -191.0\ MW + j67.0\ MVAR$$

$$S_{13} = V_1 I_{13}^* = 210.0\ MW + j105.0\ MVAR$$

$$S_{31} = V_3 I_{31}^* = -205.0\ MW - j90.0\ MVAR$$

$$S_{23} = V_2 I_{23}^* = -65.6\ MW - j43.2\ MVAR$$

$$S_{32} = V_3 I_{32}^* = 66.4\ MW + j44.8\ MVAR$$

CAPITULO III

PROGRAMACIÓN NO LINEAL BASADO EN MATLAB

3.1 Programación no lineal

El modelo de Programación No Lineal (PNL) se puede definir como aquel donde las variables de decisión se expresan como funciones no lineales ya sea en la función objetivo y/o restricciones de un modelo de optimización, es una parte de la investigación operativa cuya misión es proporcionar una serie de resultados y técnicas tendentes a la determinación de puntos óptimos para una función (función objetivo) en un determinado conjunto (conjunto de oportunidades), donde tanto la función objetivo, como las que interviene en las restricciones que determinan el conjunto de oportunidades pueden ser no lineales.

Otro concepto de Programación no lineal (PNL) se puede mencionar que es el proceso de resolución de un sistema de igualdades y desigualdades sujetas a un conjunto de restricciones sobre un conjunto de variables reales desconocidas, con una función objetivo a maximizar, cuando alguna de las restricciones o la función objetivo no son lineales[5].

Como respuesta a dos de las críticas más frecuentes de la programación lineal, a saber, la restrictividad de la hipótesis de linealidad y la dificultad de definir una única función objetivo, surge la Programación no lineal (PNL)[6].

Efectivamente, un supuesto importante en programación lineal es que todas sus funciones (objetivo y restricciones) son lineales. Aunque, en esencia, este supuesto se cumple en el caso de muchos problemas prácticos, con frecuencia no es así. Por lo que es necesario abordarlo desde la Programación No Lineal (PNL).

De manera general, un problema de PNL consiste en encontrar x = $(x_1, x_2, \ldots, x_n)$ tal que

$$\max \ f\ (x)$$

$$\text{s.a} \quad g_i(x)\ \leq \forall_i = 1,2, \ldots \ldots, m$$

$$x\ \geq 0$$

Donde $f\ (x)$ y $g_i(x)$ son funciones de n variables de minimización

$$\min \ f\ (x)$$

s.a $g_i(x) \geq b_i, \forall_i = 1,2, \ldots \ldots, m$

$x \geq 0$

En notación extendida

max $f(x_1, x_2, \ldots, x_n)$

s.a $g_1\big(f(x_1, x_2, \ldots, x_n)\big) \leq b_1$

$g_2\big(f(x_1, x_2, \ldots, x_n)\big) \leq b_2$

.

.

.

$g_m\big(f(x_1, x_2, \ldots, x_n)\big) \leq b_m$

$x_j \geq 0, \forall_j = 1,2, \ldots, n$

Existen muchos tipos de problemas de PNL, en función de las características de estas funciones, por lo que se emplean varios algoritmos para resolver los distintos tipos. Para ciertos casos donde las funciones tienen formas sencillas, los problemas pueden resolverse de manera relativamente eficiente. En algunos otros casos, incluso la solución de pequeños problemas representa un verdadero reto [7].

Cuando un problema de PNL tiene solo una o dos variables, se puede representar en forma gráfica. Si las funciones no son lineales, dibujaremos curvas en lugar de rectas, por lo que función objetivo y región factible dejaran de tener el aspecto que adquieren en la PL. La solución no tiene por qué estar en un vértice de la región factible, ni siquiera tiene porque encontrarse en la frontera de esta. Por lo que la desaparece ahora la gran simplificación que se utiliza en PL, que permite limitar la búsqueda de solución óptima a las soluciones en los vértices. Además en PNL un máximo local no es necesariamente un máximo global y en general, los algoritmos de PNL no pueden distinguir cuando se encuentra en un ´optimo local o en uno global [30]. Por tanto, es crucial conocer las condiciones en las que se garantiza que un máximo local es un máximo global en la región factible. Hay que recordar que en Análisis Matemático, cuando se maximiza una función ordinaria (doblemente diferenciable) de una sola variable f(x) sin restricciones, esta garantía está dada cuando

$$\forall_x, \frac{d^2 f}{dx^2} \leq 0,$$

Es decir, cuando la función es cóncava hacia abajo, o simplemente cóncava.

Definición

Expresando un problema de programación no lineal PNL de la siguiente manera

Encuentre los valores de las variables que

$\boldsymbol{Z = f\ (x_1, x_2, \dots, x_n)}$ **Máximo (o mínimo)**

Sujeto a $g_1\ (x_1, x_2, \dots, x_n)\ \{\leq; =; \geq\} b_1$ (1)

$g_2\ (x_1, x_2, \dots, x_n)\ \{\leq; =; \geq\} b_2$

$\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots$

$\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots$

$g_x\ (x_1, x_2, \dots, x_n)\ \{\leq; =; \geq\} b_n$

Como en la programación lineal z es el funcional del problema de programación no lineal y

$$g_1\ (x_1, x_2, \dots, x_n)\ \{\leq; =; \geq\} b_1;\ g_2\ (x_1, x_2, \dots, x_n)\ \{\leq; =; \geq\} b_2\ ; \dots;\ g_x\ (x_1, x_2, \dots, x_n)\ \{\leq; =; \geq\} b_n$$

Son las restricciones del problema de programación no lineal.

Un problema de programación no lineal es un problema de programación no lineal no restringido.

El conjunto de puntos, tal que es un número real, es, entonces, es el conjunto de los números reales.

Los siguientes subconjuntos de (llamados intervalos) serán de particular interés:

$[a, b]$ Las x que satisfacen $a \leq x \leq b$

$[a, b)$ Las x que satisfacen $a \leq x < b$

$(a, b]$ Las x que satisfacen $a < x \leq b$

(a, b) Las x que satisfacen $a < x < b$

$[a,\infty)$ = las x que satisfacen $x \geq a$

$(-\infty, b]$ = las x que satisfacen $x \leq b$

Y en forma análoga a las definiciones de la programación lineal.

Definición

La región factible para el problema de programación no lineal es el conjunto que satisfacen las m restricciones de (1)

Definición

Cualquier punto x en la región factible, para l cual se tiene que f(x) $\geq$ f(x) para todos los puntos X de la región factible, es una solución para el problema de programación no lineal

Por supuesto, si son funciones lineales, entonces (1) será un problema de programación lineal y puede resolverse mediante el algún algoritmo definido [28].

Tipos de Problemas de Programación no Lineal.

Si f (x) es diferenciable, la condición necesaria para que x=x* sea optima

$$\frac{\delta_f}{\delta_{x_j}} \downharpoonright_{x=x^*} = 0, \forall_j = 1,2,\dots,n.$$

La condición suficiente es que f(x) sea cóncava

- **Optimización restringida linealmente**: si todas las funciones de restricción son lineales pero la función objetivo es no lineal. Se han desarrollado extensiones del método simplex. Un caso particular, con m= 0 es aquel que hay variables no negativas[8].
Por ejemplo

$$\max f(x)$$
$$\text{s.a } x_j \geq 0$$

Entonces la condición necesaria cambiara a :

$$\begin{cases} \frac{\delta_f}{\delta_{x_j}} \downharpoonright_{x=x^*} = 0, & x_j^* > 0 \\ \frac{\delta_f}{\delta_{x_j}} \downharpoonright_{x=x^*} \leq 0, & x_j^* > 0 \end{cases}$$

- **Programación cuadrática:** problema restringido linealmente con función objetivo cuadrática contiene cuadrados de variables o/y productos variables de objetivo cuadrática [42].

$$f(x_1, x_2) = a_1 x_1^2 + a_2 x_2^2 + a_3 x_1 x_2 + a_4 x_1 + a_5 x_2$$

Se han desarrollado muchos algoritmos para f(x) cóncava, esta formulación surge de manera natural en muchas aplicaciones.

Programación convexa: abarca una amplia clase de problemas, entre los cuales, como casos especiales, se puede mencionar el tipo anterior cuando $f(, x_1, x_2)$ es una función cóncava que debe maximizarse.

Los supuestos son: (i) f (x) es cóncava y (ii) cada g_i(x) es convexa.
Estos supuestos aseguran que un máximo local es global.

- **Programación separable:** es un caso especial de programación convexa, en donde el supuesto adicional es: todas las funciones f (x) y g_i(x) son separables [49].

Una función separable es una función en la que cada término incluye una sola variable, por lo que la función se puede separar en una suma de funciones de variables individuales. Por ejemplo, si f (x) es una función separable, se puede expresar como

$$f_x = \sum_{j=1}^{n} f_j(x_j)$$

Donde cada $f_j(x_j)$ incluye solo los términos con x_j .
Es importante distinguir estos problemas de otros de programación convexa, pues cualquier problema de programación separable se puede aproximar muy de cerca mediante uno de programación lineal y, entonces, se puede aplicar el eficiente método simplex.

- **Programación no convexa**: incluye todos los problemas de PNL que no satisfacen los supuestos de programación convexa. En este caso, aun cuando se tenga éxito en encontrar un máximo local, no hay garantía de que sea también un máximo global. Por lo tanto, no se cuenta con un algoritmo que garantice encontrar una solución óptima para todos estos problemas; sin embargo, existen algunos algoritmos bastante adecuados para encontrar máximos locales, en especial cuando las formas de las funciones no lineales no se desvían demasiado de aquellas que se supuso para programación convexa. Ciertos tipos específicos de problemas de programación no convexa se pueden resolver sin mucha dificultad mediante métodos especiales.

- **Programación geométrica:** cuando se aplica PNL a problemas de diseño de ingeniería, muchas veces la función objetivo y las funciones de restricción toman la forma

$$g\,(x) = \sum_{k=1}^{N} C_k P_k(x)$$

Donde

$$P_k(x) = x_2^{ak1} x_2^{ak2} \dots x_n^{a_{kn}}, k\ = 1,2,\dots,N$$

3.2 Consideraciones necesarias para la programación no lineal.

Se puede dar una definición de programación no lineal PNL, por contraposición con la programación lineal PL. Hay que recordar que la programación lineal trata el problema de optimizar una función lineal $f: R^n \rightarrow R$ sujeta a una serie de restricciones también lineales.
Si el problema se modifica, cambiando la función objetivo y/o, al menos, una de las restricciones, por no lineales, se cae en el campo de la Programación No Lineal[9].

La teoría clásica de la optimización acude al empleo del cálculo diferencial para determinar los puntos en los cuales la función $f: R^n \rightarrow R$ asume valores óptimos (máximos y mínimos) [31].

Programación cuadrática

La programación cuadrática es un caso particular de la programación no lineal en el que la función objetiva es cuadrática y las restricciones son lineales.
Max (min) z = $C^T X + X^T DX$
St
AX $\leq b$
X ≥ 0
La función $X^T DX$ define una forma cuadrática donde D es simétrica. La matriz, se define:
$>$ 0 definida positiva (mínimo)
≥ 0 semidefinida positiva (cuasimínimo)
$<$ 0 definida negativa (máximo)
≤ 0 semidefinida negativa (cuasimáximo)
$= 0$ indefinida (puntos de inflexión)
Tendremos un mínimo si los menores principales asociados con D son todos positivos y un máximo si los menores principales tienen el signo $(-1)^i$.

Programación separable.

A los problemas de programación no lineal se les denomina de programación separable cuando las variables de decisión aparecen en términos separables tanto en función objetivo como en las restricciones.

Max (min) $\quad z = \sum_{j=1}^{n} f_j(x_j) = f_1(x_1) + f_2(x_2) + \cdots + f_n(x_n)$

St $\quad \sum_{j=1}^{n} g_{ij}(x_j) \leq b_i \ (i = 1,2 \ldots, m)$

Programación Estocástica

La programación aleatoria o estocástica se presenta cuando algunos o todos los parámetros del problema se pueden describir mediante variables aleatorias.

Nuestro objetivo es expresar la naturaleza probabilística del problema en términos determinísticos.

Dado el problema de programación no lineal en forma canónica

Max z = $\sum_{i=1}^{n} c_i \, x_i$

St $\sum_{j=1}^{n} a_{ij} x_j \leq b_i$

$x_j \geq 0 \;\; \forall_j = 1,2, \dots, n$

Supondremos que los coeficientes del problema son variables aleatorias que siguen distribuciones normales.

3.3 Estudio de Flujo de potencia óptimo.

Optimización de Flujo de carga (OPF, por sus siglas en inglés) se pude definir como la determinación de los calores de las variables de control en una red de transmisión de energía eléctrica tomando en cuenta varias restricciones que generalmente no son lineales[10].

Son utilizadas varias técnicas para realizar los cálculos para resolver un OPF.

Tenemos un ejemplo, la técnica Newton Rapshon se la podría utilizar en la subrutina de Flujo de Carga, mientras que métodos de análisis de sensibilidad, programación lineal o programación no lineal son utilizados para la optimización.

Cuando se está en esta etapa del proceso, el objetivo principal es la minimización de alguna función sujeta a una serie de restricciones lineales o no, definidas por inecuaciones y ecuaciones.

Cuando se trata de una ecuación no lineal del problema requiere un proceso iterativo, y en algunos casos pueden ser necesarios ajustes de variables de control para lograr una aproximación a la solución óptima de manera eficiente [50].

De los procedimientos de optimización, resultan nuevos valores para dichas variables a través de la solución de un flujo de carga para las nuevas condiciones de operación del sistema, el cual determinara el perfil de tensiones óptimas del mismo.
La optimización del flujo de potencia puede ser descrita como sigue:

Minimizar	*función (x,u)*	
Sujeto a	$h(x,u)= 0$	
	$g(x,u)\geq 0$	*Ecuación (1)*

Donde

u = conjunto de variable de control
x = variables dependientes
f = la función objetivo a minimizar
h, g = conjunto de restricciones de igualdad (ecuaciones de la red) y desigualdad (limites en las variables) respectivamente[16].

Resulta algo complejo, describir el estado óptimo de un sistema eléctrico mediante una función, esto se debe principalmente a los diferentes factores que pueden ser considerados importantes en la solución.
Para algunos sistemas eléctricos se explotan de la forma más económica posible, imponiendo restricciones únicamente sobre el estado real de la red, otros, son explotados en modo preventivo, de forma que las magnitudes eléctricas permanezcan dentro de los limites incluso tras determinar contingencias.
En conclusión, lo importante de la función objetivo es que logre encontrar una solución frente a un estado de la red, gracias a esto hace que cumpla los siguientes objetivos;

Sistema en estado normal.-

Estado alerta o inseguro, por lo que la seguridad del sistema es prioritaria el objetivo será transformar el sistema ha estado seguro incurriendo en el mínimo de los costos.
Estado seguro, se desea minimizar el costo de generación, y si el costo se ha calculado previamente, lo importante es determinar el perfil óptimo de tensión minimizando así las pérdidas de producción. También, es importante actuar sobre los elementos de control del sistema, con el fin de minimizar las pérdidas de potencia activa en la red.

Sistema en estado de emergencia.-
Para este caso se debe definir el mínimo de pasos a seguir que conduzca a un estado normal en el menor tiempo posible, incluyendo criterios económicos y de eficiencia.

Cuando se el caso de que no sea posible llevar el sistema nuevamente a estado normal de forma rápida, minimizar el número de violaciones de los limites operacionales.

Concluyendo y en base a la información presentada el problema del flujo óptimo de potencia OPF, representa más que un problema matemático, constituye un algoritmo complejo que requiere la incorporación de ciertas reglas para alcanzar una solución satisfactoria.

Los sistemas de ecuaciones que involucran la optimización de flujo de potencia OPF se los puede resolver a través de diferentes herramientas y métodos numéricos como la programación lineal y no lineal [40].

3.4 Simulaciones con Matlab

3.4.1. Introducción

MATLAB es una de las muchas sofisticadas herramientas de computación disponibles en el comercio para resolver problemas de matemáticas, tales como Maple, Mathematicay MathCad. A pesar de lo que afirman sus defensores, ninguna de ellas es "la mejor". Todas tienen fortalezas y debilidades. Cada una permitirá efectuar cálculos matemáticos básicos, pero difieren en el modo como manejan los cálculos simbólicos y procesos matemáticos más complicados, como la manipulación de matrices. Por ejemplo, MATLAB es superior en los cálculos que involucran matrices, mientras que Maple lo supera en los cálculos simbólicos. El nombre mismo de MATLAB es una abreviatura de Matrix Laboratory, laboratorio matricial. En un nivel fundamental, se puede pensar que estos programas son sofisticados calculadoras con base en una computadora. Son capaces de realizar las mismas funciones que una calculadora científica, y muchas más[11].

En muchas clases de ingeniería, la realización de cálculos con un programa de computación matemático como MATLAB sustituye la programación de computadoras más tradicional.

Dado que MATLAB es tan fácil de usar, muchas tareas de programación se llevan a cabo con él. Sin embargo, MATLAB no siempre es la mejor herramienta para usar en una tarea de programación. El programa destaca en cálculos numéricos, especialmente en los relacionados con matrices y gráficas. Por lo general, los programas de alto nivel no ofrecen acceso fácil a la graficación, que es una aplicación en la que destaca MATLAB.

El área principal de interferencia entre MATLAB y los programas de alto nivel es el "procesamiento de números": programas que requieren cálculos repetitivos o el procesamiento de grandes cantidades de datos. Tanto MATLAB como los programas de alto nivel son buenos en el procesamiento de números.

Por lo general, es más fácil escribir un programa que "procese números" en MATLAB, pero usualmente se ejecutará más rápido en C++ o FORTRAN. La única excepción a esta regla son los cálculos que involucran matrices: puesto que MATLAB es óptimo para matrices, si un problema se puede formular con una solución matricial, MATLAB lo ejecuta sustancialmente más rápido que un programa similar en un lenguaje de alto nivel.

MATLAB se utiliza mucho en ingeniería eléctrica para aplicaciones de procesamiento de señales.

Así como también en las disciplinas de ciencias y programación de computadoras, es importante tener un enfoque consistente para resolver los problemas técnicos. El enfoque que se plantea a continuación es útil en cursos tan distintos como química, física, termodinámica y diseño de ingeniería. También se aplica a las ciencias sociales, como economía y sociología.

Para la resolución de problemas se sugiere los siguientes pasos;

- Plantear el problema.

En esta etapa con frecuencia es útil hacer un dibujo.

Si no tiene una comprensión clara del problema, es improbable que pueda resolverlo.

- Describir los valores de entrada (conocidos) y las salidas (incógnitas) que se requieren.

Se debe tener cuidado de incluir las unidades conforme describe los valores de entrada y salida. El manejo descuidado de las unidades con frecuencia lleva a respuestas incorrectas.

Identifique las constantes que tal vez requiera en el cálculo, como la constante de los gases ideales y la aceleración de la gravedad.

Si es apropiado, en un dibujo escriba los valores que haya identificado o agrúpelos en una tabla.

- Desarrollar un algoritmo para resolver el problema. En aplicaciones de cómputo, es frecuente que esto se logre con una prueba de escritorio

Para ello necesitará.

Identificar cualesquiera ecuaciones que relacionen los valores conocidos con las incógnitas.

Trabajar con una versión simplificada del problema, a mano o con calculadora.

- Resolver el problema.

Probar la solución

¿Sus resultados tienen sentido físico?

¿Coinciden con los cálculos de la muestra?

¿La respuesta es la que se pedía en realidad?

Las gráficas con frecuencia son formas útiles de verificar que los cálculos son razonables.

3.4.2 Comparación con Software actuales

A continuación se detalla un aproximado comparativo de los paquetes de software actuales que se basan en la plataforma Matlab para el análisis y simulación de sistemas de potencia.

PAQUETE	EST	MatEMTP	Matpower	PAT	PSAT
PF	•		•	•	•
CPF					•
OPF			•		•
SSSA	•			•	•
TDS	•	•		•	•
EMT		•			
GIU		•			•
CAD	•	•		•	•

Tabla 2.2 ***Paquetes basados en MATLAB para el análisis de sistemas eléctricos de potencia*** **[56].**

Donde:

PF.- Flujo Óptimo de Potencia
CPF.- Continuación de flujo de potencia y/o análisis de estabilidad de voltaje
OPF.- Flujo óptimo de potencia (Optimal Poer Flow)
SSSA.- Análisis de estabilidad de pequeña señal (Samll Signal Stability Analysis)

TDS.- Simulación en el dominio del tiempo y transitorios electromagnéticos (Time Domain Simulation) y (Electromagnetic Transient) respectivamente
GUI.- Interfaz gráfica del usuario (Graphical User Interface)
CAD.- Construcción de red grafica
Paquetes:

EST. - Educational Simulation Tool
MatEMTP.-
Matpower
PST. - Power System Toolbox
PAT. - Power Anlysis Toolbox
SPS. - SimPowerSystems
VST. - Volatage Stability Toolbox

3.5 PSAT (Power System Analysis Toolbox)

3.5.1 Gui Principal

Una interfaz gráfica es el vínculo entre el usuario y un programa computacional, constituida generalmente por un conjunto de comandos o menús, instrumentos y métodos por medio de los cuales el usuario se comunica con el programa durante las operaciones que se desean realizar, facilitando la entrada y salida de datos e información[12] .

Una interfaz es una de las partes más importantes de cualquier programa puesto que determina qué tan factible y preciso será el desempeño del programa ante los comandos que el usuario pretenda ejecutar. Aunque un programa sea muy poderoso, si se manipula por medio de una interfaz pobremente elaborada, tendrá poco valor para un usuario inexperto. Es por esto que las interfaces gráficas tienen una gran importancia para usuarios inexpertos o avanzados de cualquier programa ya que facilitan su uso.

Ejemplos de interfaces gráficas son las ventanas de Word, Excel, la ventana de Matlab entre otras. Una interfaz gráfica consta de botones, menús, ventanas, etc, que permiten utilizar de una manera muy simple y en ocasiones casi intuitiva programas realizados en ambientes como Windows y Linux. Las interfaces gráficas también se conocen como interfaces de usuario.

El nombre en inglés de las interfaces gráficas es Graphical User Interfase y se denominan GUI.

Existen diferentes lenguajes de programación que permiten crear GUIs tales como Visual C, Visual Basic, TK y MATLAB por mencionar algunos. Todos ellos permiten usar diferentes controles y tienen distintas maneras de programarlos.

MATLAB nos permite realizar GUIs de una manera sencilla usando una herramienta llamada GUIDE (GUI Development Environment).

GUIDE (Graphical User Interfase Development Environment) es un juego de herramientas que se extiende por completo en el soporte de MATLAB, diseñadas para crear GUIs (Graphical User Interfaces) fácil y rápidamente, prestando ayuda en el diseño y presentación de los controles de la interfaz, reduciendo la labor al grado de seleccionar, tirar, arrastrar y personalizar propiedades[13].

Una vez que los controles están en posición se editan las funciones de llamada (Callback) de cada uno de ellos, escribiendo el código de MATLAB que se ejecutará cuando el control sea utilizado. Siempre será difícil diseñar GUIs, pero no debería ser difícil implementarlas.

GUIDE está diseñado para hacer menos tedioso el proceso de aplicación de la interfaz gráfica y obviamente para trabajar como herramienta de trazado de GUIs. Entre sus poderosos componentes está el editor de propiedades (property editor), éste se encuentra disponible en cualquier momento que se esté lidiando con los controles de MATLAB. El editor de propiedades por separado se puede concebir como una herramienta de trazado, y asistente de codificación (revisión de nombres y valores de propiedades). Cuando se fusiona con el panel de control, el editor de menú, y herramienta de alineación, resulta una combinación que brinda un inigualable control de los gráficos en MATLAB.

El beneficio que proporciona el uso de GUIs es evidente, ya que permiten al usuario ejecutar cómodamente código desarrollado en MATLAB sin necesidad de cumplir la incómoda sintaxis funcional necesaria cuando se trabaja desde la línea de órdenes. A diferencia de la ejecución de funciones o scripts de MATLAB, la ejecución de GUIs no ejecución del código. Es el usuario, a través de su interacción con el GUI, el que determina el orden en que se ejecutan las diferentes órdenes y funciones desarrolladas. Otra diferencia importante es que la ejecución no termina cuando finaliza la ejecución del script o función, sino que el GUI permanece abierto, permitiendo al usuario invocar la ejecución de ese u otro código desarrollado. El desarrollo de GUIs se realiza en dos etapas

Diseño de los componentes (controles, menús y ejes) que formarán el GUI. Codificación de la respuesta de cada uno de los componentes ante la interacción del usuario.

Iniciando GUIDE

A la herramienta GUIDE se accede de varias maneras, la primera de ellas es tecleando guide en la ventana de comando.

>> guide

Otra manera de entrar en GUIDE es través de la opción File, haciendo clic en New y por último eligiendo la opción GUI, (como se muestra en la figura 3.1).

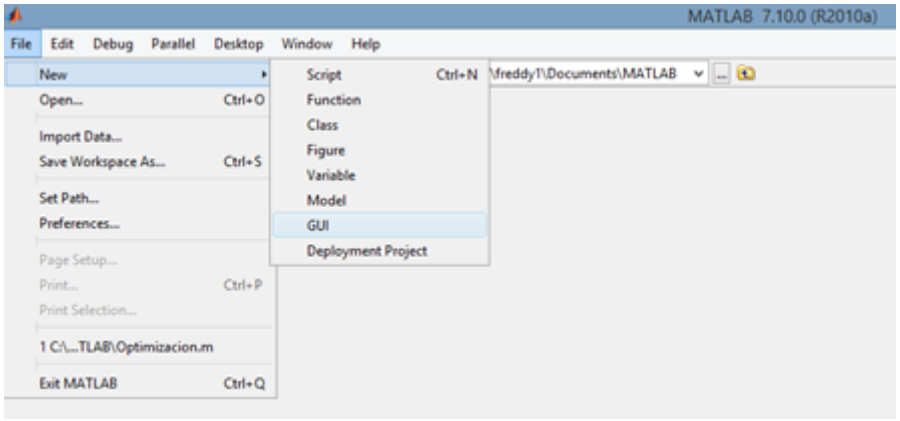

Figura 3.1 ***Ubicación inicio de un GUI***

A continuación, se presenta el siguiente cuadro de diálogo, correspondiente con la Ventana de inicio de GUI:

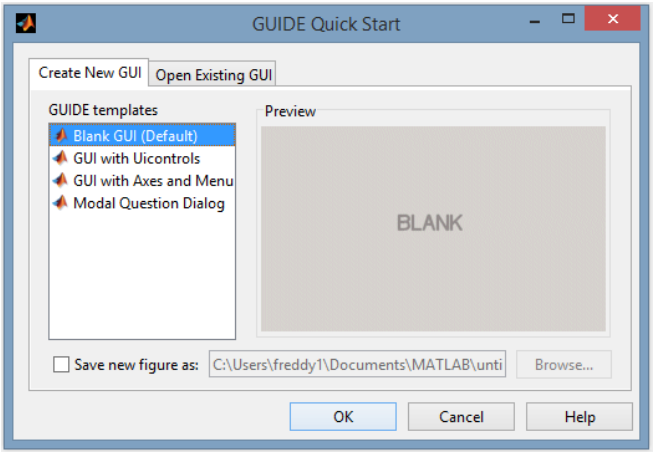

Figura 3.2 ***Ventana de inicio de GUI***

Donde se presentan las siguientes opciones:

a) Blank GUI (Default)

La opción de interfaz gráfica de usuario en blanco (viene predeterminada), nos presenta

un formulario nuevo, en el cual podemos diseñar nuestro programa.

b) GUI with Uicontrols

Esta opción presenta un ejemplo en el cual se calcula la masa, dada la densidad y el volumen, en alguno de los dos sistemas de unidades. Podemos ejecutar este ejemplo y obtener resultados.

c) GUI with Axes and Menu

Esta opción es otro ejemplo el cual contiene el menú File con las opciones Open, Print y Close. En el formulario tiene un Popup menu, un push button y un objeto Axes, podemos ejecutar el programa eligiendo alguna de las seis opciones que se encuentran en el menú despegable y haciendo click en el botón de comando.

Flujo de operación con GUI

Con un GUI, el flujo de cómputo está controlado por las acciones en la interfaz.

Mientras que en un script el flujo de comandos está predeterminado, el flujo de operaciones con una GUI no lo está. Los comandos para crear una interfaz con el usuario se escriben en un script, la interfaz invoca que se ejecute el script, mientras la interfaz del usuario permanece en la pantalla aunque no se haya completado la ejecución del script.

En la figura 3.3 se muestra el concepto básico de la operación del software con una GUI. Cuando se interactúa con un control, el programa registra el valor de esa opción y ejecuta los comandos prescritos en la cadena de invocación. Los menús de interfaz con el usuario, los botones, los menús desplegables, los controladores deslizantes y el texto editable son dispositivos que controlan las operaciones del software. Al completarse la ejecución de las instrucciones de la cadena de invocación, el control vuelve a la interfaz para que pueda elegirse otra opción del menú. Este ciclo se repite hasta que se cierra el GUI.

El control guarda un string que describe la acción a realizar, cuando se invoca puede consistir en un solo comando de MATLAB o una secuencia de comandos, o en una llamada a una función.

Es recomendable utilizar llamadas a funciones, sobre todo cuando se requieren de más de unos cuantos comandos en la invocación.

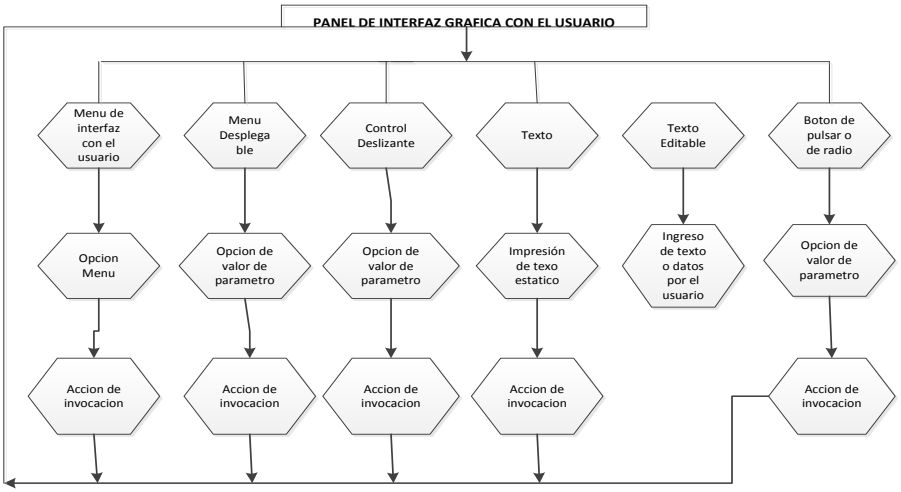

Figura 3.*3 Flujo de opciones de GUI*

Básicamente solo se necesitan entender cinco comandos para poder describir una GUI: uimenu, uicontrol, get, set y-axes. No obstante, lo que hace relativamente complicado a estos comandos es el gran número de formas de uso que tienen. Es imposible describir todos los tipos de situaciones, pues requiere demasiado espacio y sería muy laborioso leerlo.

Creación de una GUI sencilla.

Para crear una GUI sencilla primero tenemos que llamar a la ventana de comandos de MatLab. Esto se hace picando sobre el icono de MatLab, figura 3.4, que al abrirse nos mostrará la siguiente ventana:

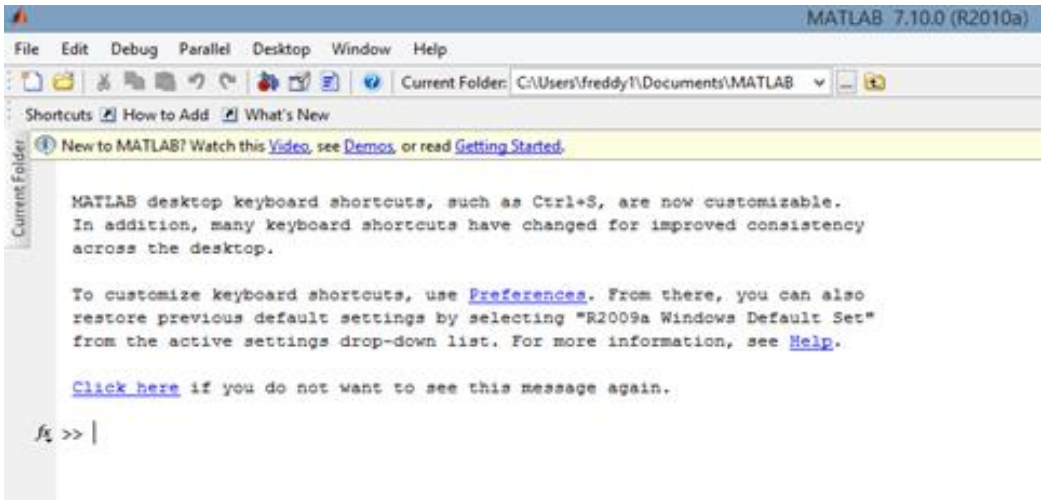

Figura 3.4 *Ventana de comandos Matlab*

Al seleccionar que queremos una GUI se nos abre una ventana de diálogo en la que tenemos una serie de opciones.

Podemos abrir una GUI que ya exista, lo cual haremos después de haber guardado

nuestra primera GUI o comenzar con una GUI en blanco. Las otras ofrecen una plantilla en la que ya existen algunos controles (Uicontrols), o en la que ya hay incorporado un gráfico y un menú y finalmente incorporar un ventana de diálogo.

En la figura 3.5, observamos un Check Box que dice Save on startup as: y una ventana que permite seleccionar un nuevo directorio. Se marca la casilla para activar la opción Browse e introduce el nombre de una interfaz en un directorio en el que no haya nada. El nombre que se eligió es untitled en un directorio nuevo y se tiene lo siguiente:

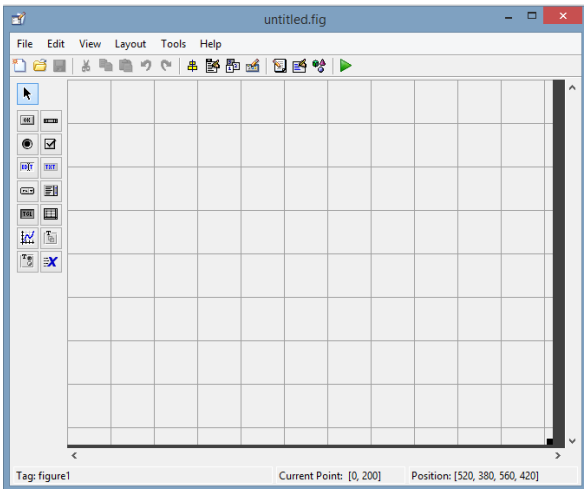

Figura 3.5 ***Ventana de programación de GUI***

Como se puede observar, el nombre de la interfaz es el elegido. Como no se ha guardado, MatLab la ha titulado provisionalmente a la interfaz como untitled. Figura 3.5 y la ha guardado en el directorio donde se hallaba la ventana de comandos.

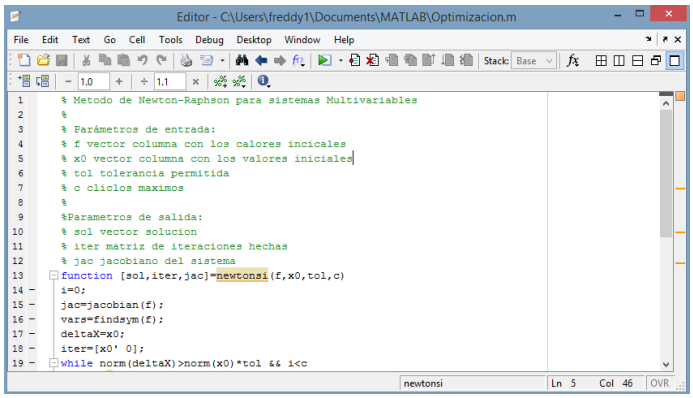

Figura 3.6 ***Ventana de comandos***

Lo que vemos aquí es el archivo de tipo *function* con la extensión característica de MatLab .m. Este archivo es construido automáticamente por MatLab y las líneas de código que aparecen son las que crean la interfaz que aparece en el archivo

optimización.m de la figura 3.6. El archivo de inicialización define los parámetros básicos de la interfaz y crea un conjunto de handles para cada uno de los objetos que vayan apareciendo sobre la interfaz.

Si hacemos clic sobre el archivo optimización.m, en la ventana de comandos obtenemos lo siguiente:

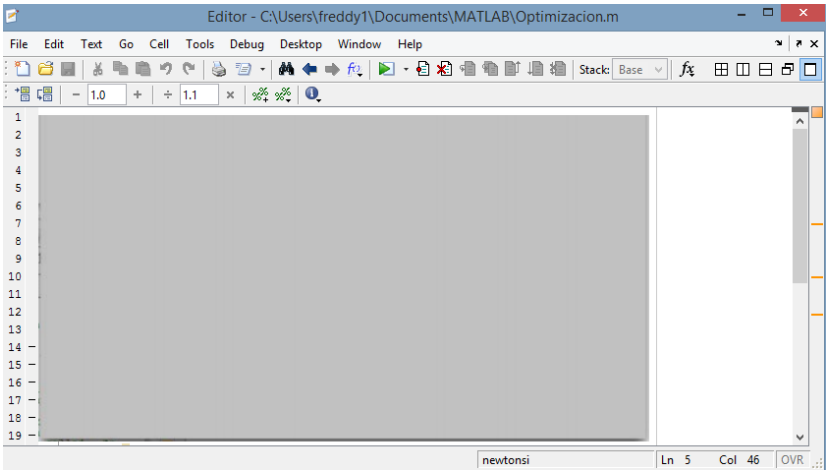

Figura 3.7 ***Interfaz de GUI***

Como se puede apreciar, la interfaz no contiene nada más que un lienzo vacío sin ninguna funcionalidad. El código de inicialización que está en este momento escrito en optimización.m se encarga de crear la interfaz tal y como está, y por esa razón ese código no debe ser modificado.

3.5.2 Modelos Definidos por el Usuario

Las funciones definidas por el usuario son una serie de bloques disponibles en la librería Simulink/User-Defined Functions que permiten aumentar la funcionalidad del modelo creado, gracias a que posibilitan trasladar funciones y código perteneciente de MATLAB y otros lenguajes al modelo en Simulink, de forma escrita y sencilla.

Fcn

El bloque Fcn, que se muestra en la figura 3.8 donde se pueden ver sus parámetros de configuración, aplica una expresión matemática definida por el usuario que es función de la entrada.

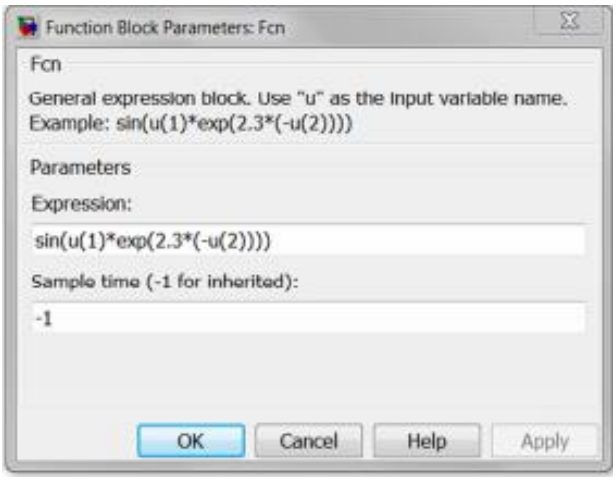

Figura 3.8 ***Bloque de la función definida por el usuario Fcn***

La expresión difiere respecto a una expresión de MATLAB en que esta no puede realizar cálculos de matrices. Además de no ser compatible con el operador dos puntos (:). Así, la entrada a un Bloque Fcn puede ser un escalar o un vector, sin embargo, la salida es siempre un escalar. Si lo que se desea es obtener un vector de salida se puede utilizar un Function block para ello:

Se permite hacer uso en la expresión de:

Constantes numéricas.

Operadores aritméticos (+ , - , * , / , ^).

Operadores relacionales (== , => , <> , =< , =!). La expresión devuelve 1 si la relación es verdadera, de lo contrario, devuelve 0.

Operadores lógicos (!, && , | |). La expresión devuelve 1 si la relación es verdadera, de lo contrario, devuelve 0.

Paréntesis.

Funciones matemáticas (abs, acos, asin, atan, atan2, ceil, cos, cosh, exp, fabs, floor, hypot, ln, log, log10, pow, power, rem, sgn, sin, sinh, sqrt, tan y tanh).

Las variables del Workspace.

MATLAB Function

Con un bloque MATLAB Function se puede escribir una función de MATLAB para ser usada en un modelo de Simulink. La función de MATLAB creada es ejecutada durante la simulación y genera código para Simulink.

Al contrario de lo que pasaba con las Fcn, ahora se puede especificar el número de entradas y salidas necesarias para implementar la función de MATLAB en el encabezado de esta, como argumentos y valores devueltos según se puede observar en la figura 3.9, además se tiene la posibilidad de que tanto las entradas como las salidas sean en forma matricial.

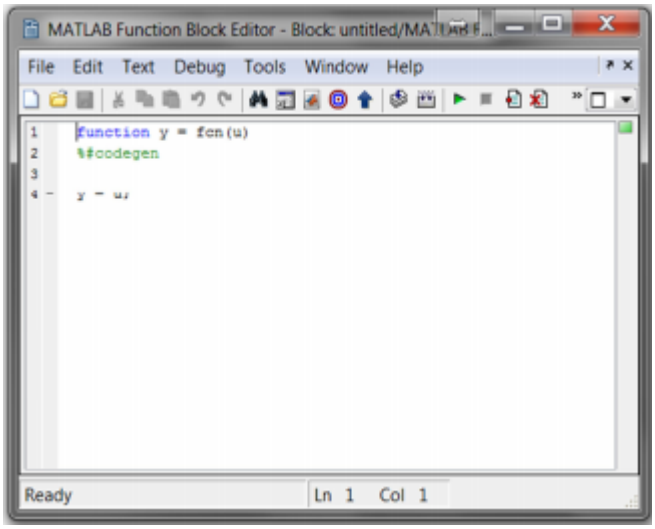

Figura 3.9 ***Bloque de la función definida por el usuario MATLAB Function***

Este bloque puede generar un código embebido eficiente, que se basa en el análisis del tamaño, clase y complejidad de cada variable. Sin embargo este análisis impone ciertas restricciones:
La primera asignación de una variable debe definir su tamaño, clase y complejidad.
Por lo general, no se pueden reasignar propiedades a las variables después de la asignación inicial, excepto cuando se utilizan datos de tamaños variables o se reutilizan las variables en el código para diferentes propósitos.
Además de estas restricciones de tipo lingüísticas, este bloque de MATLAB sólo admite un subconjunto de las funciones de las que dispone. Estas se incluyen en categorías comunes tales como:

- Funciones aritméticas como plus, minus y power.
- Operaciones con matrices como size y length.
- Operaciones avanzadas de matrices como lu, inv, svd y

Aunque el código de este bloque trata de producir exactamente los mismos resultados que se obtendrían con MATLAB, las diferencias pueden deberse a errores de redondeo, que pueden ser de unos pocos eps inicialmente para ir aumentando con el paso del tiempo tras realizarse un gran número de operaciones repetidas. Esta variable eps, es la mínima distancia que hay entre la representación de un número y el siguiente más cercano, y surge debido a la naturaleza discreta de los sistemas computacionales, con lo que en esencia el eps es el mínimo error de precisión

que se está cometiendo. Además la confianza en el comportamiento de los NaN (Not a Number), por ejemplo, no es recomendable. Todo esto se une al hecho de que diferentes compiladores de C pueden producir resultados distintos para el mismo cálculo.
Las llamadas recursivas no están permitidas en los bloques de funciones de MATLAB

Function

Una S-Function (System-Functions) es un lenguaje descriptivo escrito dentro de un bloque de Simulink en MATLAB, C, C++ o Fortran. Para C, C++ y Fortran las S-Function se compilan como archivos MEX haciendo uso de mex utility. Al igual que ocurre con otros archivos MEX, las S-Function se vinculan dinámicamente a subrutinas que el intérprete de MATLAB puede cargar automáticamente y ejecutar. En la ventana de configuración que se muestra en la
Figura 3.10 se puede ver que se necesitan unos parámetros de base para crear la S-Función, que se pueden introducir a través de una máscara, y el archivo que contenga el código de esta.

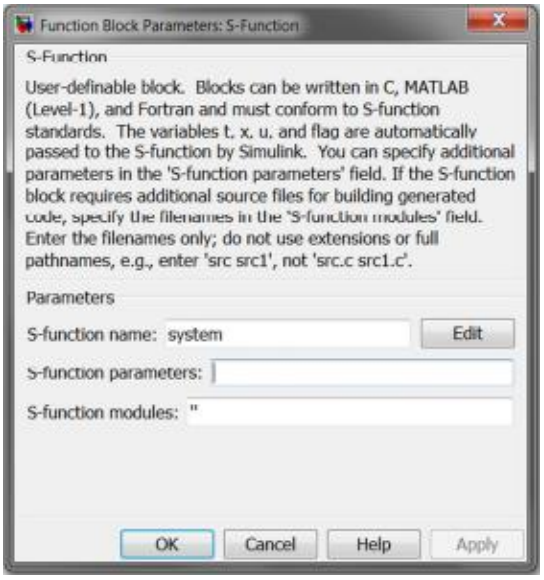

Figura 3.10 ***Bloque de la función definida por el usuario S-Function***

Las S-Function utiliza una sintaxis de llamada especial denominada S-Function API, que le permite interactuar con el motor de Simulink. Esta interacción es muy similar a la interacción que tiene lugar entre el motor y el sistema de bloques de Simulink
Las S-Function son un poderoso mecanismo que permite ampliar las capacidades de Simulink, pudiéndose integrar estas funciones en sistemas continuos, discretos e híbridos. Siguiéndose una serie de pautas simples se puede implementar un algoritmo haciéndose uso de una S-Funtion que puede posteriormente ser usada como un bloque en un modelo distinto de Simulink.

En todo momento se tiene la posibilidad de personalizar el código generado para las S-Funtion escribiéndose el archivo TLC(Target Language Compiler) y está permitido el uso de máscaras para crear cuadros de diálogo personalizados e iconos para el bloque de la S-Function. Los cuadros de diálogos de las máscaras pueden hacer que sean más fáciles especificar parámetros adicionales.

Las S-Function son de gran utilidad para un amplio número de aplicaciones, entre las que se incluyen:

- Creación de nuevos bloques de propósito general.
- Añadir bloques que representen controladores de dispositivos físicos.
- Incorporar en la simulación, código ya existente de C.
- Describir un sistema como un conjunto de ecuaciones matemáticas.
- Uso de animaciones gráficas.

El uso más común de las S-Function es la primera, crear bloques de Simulink que se puedan usar muchas veces, en uno o varios modelos, modificando únicamente los parámetros con cada instancia del bloque.

Para crear las S-function es necesario entender cómo trabajan este tipo de funciones. Tal conocimiento requiere una comprensión de cómo el motor de Simulink simula un modelo, incluyendo las matemáticas de los bloques.

Un bloque de Simulink se compone de un conjunto de entradas, estados y salidas, como se puede ver en la figura 3.11, donde las salidas son función del tiempo de simulación, las entradas y los estados.

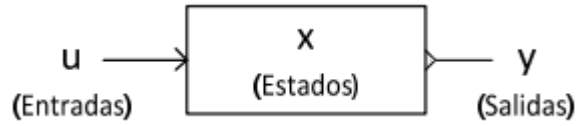

Fig. 3.11 ***Diagrama de la relación de entradas, estados y salidas en un modelo de Simulink*** [62].

De esta forma se puede representar las relaciones matemáticas entre las entradas, salidas, estados, y el tiempo de simulación con las siguientes expresiones:

$$\begin{cases} y = f_o(t,x,u) & (Output) \\ x = f_d(t,x,u) & (Derivatives) \\ x_{d_{k+1}} = f_u(t,x_c,x_d,u) & (Update) \end{cases}$$

Donde:

$$x = [x_c; x_d] \qquad 1.2$$

A partir de aquí la ejecución del modelo procede por etapas. En la Figura 3.12 se ilustra las etapas de una simulación.

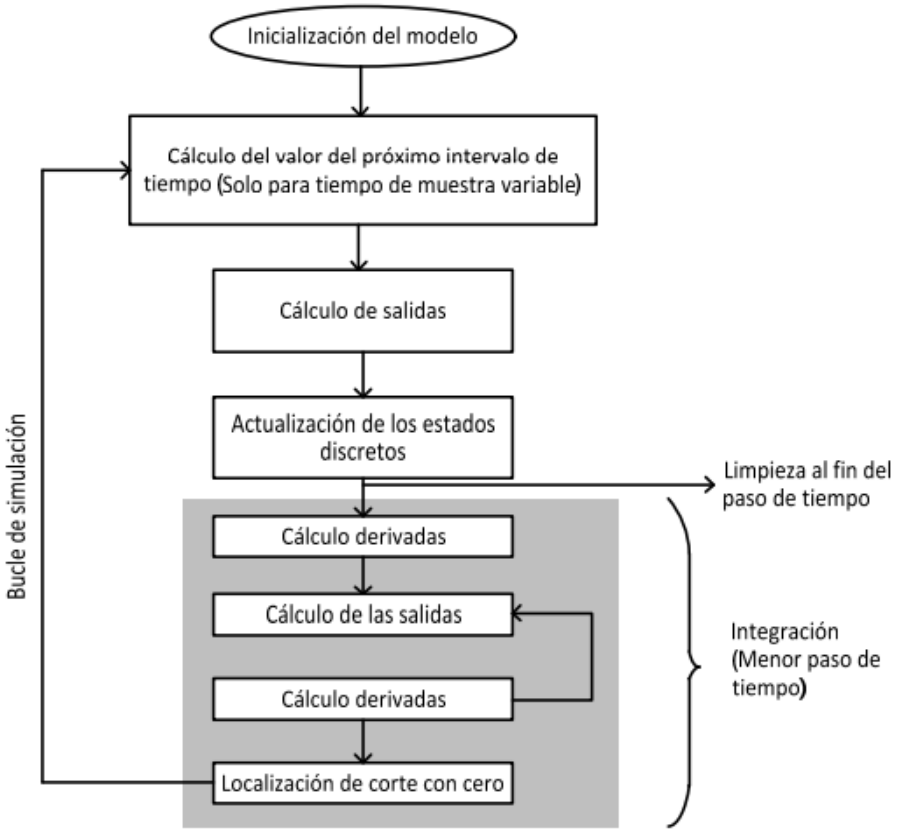

Fig. 3.12 ***Etapas de ejecución de una S-Function*** **[60].**

Durante la simulación de un modelo, en cada etapa, el motor de Simulink llama a los métodos apropiados para cada uno de los bloque de S-Function contenidos en él. Las tareas desempeñadas por las llamadas a métodos de las S-Function incluyen:

- Inicialización
- Calculo del siguiente paso del tiempo
- Calculo de las salidas en el paso del tiempo actual
- Actualización de estados discretos en el paso del tiempo actual
- Integración

Function Builder

El bloque S-function Builder crea una C MEX S-Function a partir de las especificaciones que se le definen en el constructor y el código fuente en C que se le proporcione. En la figura 3.13 se muestra el bloque de esta función definida por el usuario y su ventana de configuración

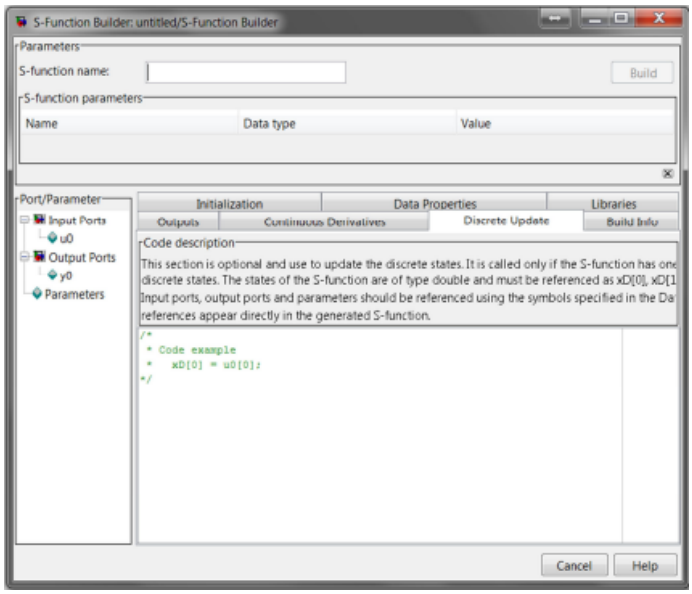

Fig. 3.13 ***Bloque constructor de las S-Funciones, S-Function Builder***

Las instancias del bloque S-Function Builder también sirve como contenedor para las S-Function generadas en los modelos Simulink. Cuando se produce la simulación de un modelo que contiene las instalaciones de un bloque S-Funcion Builde, el software de Simulink invoca la S-Function generada asocia para calcular la salida de la instancia en cada paso de tiempo.

El bloque generador de S-Function no es compatible con el uso de enmascaramiento. Sin embargo sí que es posible usarse una máscara en un bloque de subsistema que contenga un bloque S-Function Builder.

3.5.3 Modelos y Rutinas PSAT

PSAT es una herramienta de código abierto basada en MATLAB para análisis y control de sistemas de potencia. La versión de línea de comando del PSAT también es compatible con GNU Octave (homólogo de MATLAB para Linux). Puede ser utilizado

en gran variedad de sistemas de potencia: desde pequeñas redes para propósitos académicos hasta sistemas reales de tamaño medio.

PSAT puede realizar flujos de potencia, flujos de potencia continuos, flujos óptimos de potencia, análisis de estabilidad de pequeña señal y simulaciones en el dominio del tiempo. Mediante una interfaz gráfica de usuario (GUIs) y una librería basada en Simulink se pueden ejecutar todas las órdenes de forma sencilla[14].

El principal objetivo de la herramienta UDM es extender las capacidades del PSAT y asistir a los usuarios finales con poca habilidad de programación para construir y configurar sus propios modelos. La UDM está disponible solo en la plataforma Matlab ya que esta hace uso de herramientas matemáticas simbólicas.

En la tabla I enlistamos algunos modelos de simulación típicos de PSAT

MODELO	ELEMENTOS
FLUJO DE CARGA	Buses de compensación, PV y PQ; líneas de transmisión, transformadores, generadores, cargas constantes y admitancias en paralelo
MERCADO	Oferta de generación con límites, reservas de generación y demanda
PROTECCIONES	Interruptores y protectores de líneas
PMU's	Medidores de frecuencia de buses
CARGAS	Cargas dependientes del voltaje y de frecuencia, ZIP cargas no lineales polinomiales y exponenciales, cargas térmicamente controladas.
MÁQUINAS	Máquinas Síncronas y motores de inducción
CONTROLES	Gobernadores de turbina, AVR's, PSS's, límites de sobre excitación y regulación secundaria de voltaje
REGULACION DE TRANSFORMADORES	Cambiadores de tap y transformadores de fase cambiada
FACTS	SVC's, TCSC's, SSSC's, UPFC's
TURBINAS	Turbinas de viento de velocidad constante con motor de inducción de jaula de ardilla, turbinas de viento de velocidad variable con generadores síncronos
OTROS MODELOS	Máquinas síncronas con ejes dinámicos, sub síncronas de resonancia, celdas de óxido completo y áreas equivalentes de redes de transmisión

Tabla I ***Modelos típicos de simulación de PSAT***

El primer paso es ingresar en el GUI del UDM las variables y el sistema de ecuaciones diferenciales algebraicas que describen el nuevo modelo. PSAT compila automáticamente las ecuaciones, calcula las expresiones simbólicas de las matrices Jacobianas y expresa una función de Matlab para el nuevo componente. El usuario puede guardar el modelo y/o instalarlo en PSAT, el UDM también tiene un des instalador del modelo, así, cuando el modelo es innecesario este puede ser desinstalado sin ningún problema.

PSAT tiene una GUI con opciones de configuración para elementos y algoritmos (por

ejemplo, determinar el número máximo de iteraciones del algoritmo de Newton-Raphson), pero delega el manejo de datos a variables globales que permiten el manejo de comandos de línea, necesarios en muchas ocasiones. Mientras el soporte para la GUI de PSAT mostrada en la Figura 3.14, es el lenguaje de Matlab, Simulink puede considerarse también como un ambiente gráfico a este programa. Con una librería gráfica de Simulink expuesta en la Figura 3.15, los modelos de PSAT pueden construir topologías de redes y extraer datos de componentes de forma cómoda, sin embargo, esta GUI no está disponible en GNU/Octave; por lo que el análisis de redes se ve limitada a líneas de comando.

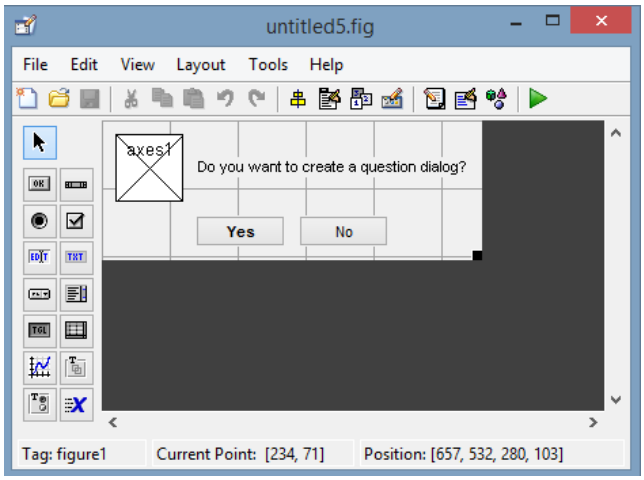

Fig. 3.14 ***Soporte para GUI***

A continuación se muestra la librería que contiene Simulink

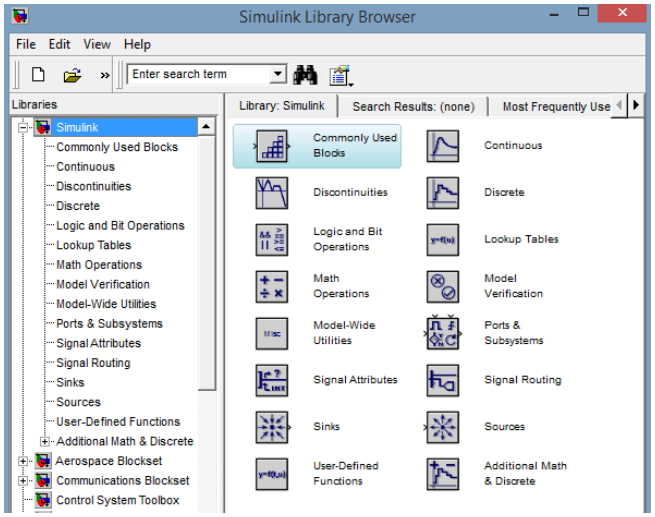

Fig. 3.15 ***Librería que presenta Simulink en Matlab***

Modelos matemáticos de PSAT

Algoritmo de flujo de carga

PSAT utiliza un sistema de ecuaciones no lineales para definir el modelo básico de un sistema de potencia [46].

$$\begin{bmatrix} X_c \\ P_c \\ Q_c \end{bmatrix} = \begin{bmatrix} f_c(X_c, Y_c, P_c) \\ f_{pc}(X_c, Y_c, P_c) \\ f_{qc}(X_c, Y_c, P_c) \end{bmatrix} \qquad 1.1$$

Donde

X_c representa las variables de estado
Y_c representa las variables algebraicas
P_c representa las variables independientes del sistema
f es una ecuación diferencial
g una ecuación algebraica definida por

$$g(x, y, p) = \begin{bmatrix} g_p \\ g_q \end{bmatrix} = \begin{bmatrix} g_{pm} \\ g_{qm} \end{bmatrix} - \sum \begin{bmatrix} g_{pb} \\ g_{qc} \end{bmatrix} \qquad 1.2$$

Donde;

g_{pm} y g_{qm} representa la potencia trasmitida por el bus m y

g_{pc} y g_{qc} la potencia generada

El algoritmo de Newton-Raphson es el método que PSAT utiliza para dar solución al flujo de carga de los sistemas de potencia, linealizando la ecuación 1.1 si no se conocen los parámetros de entrada, por ejemplo el voltaje y torque de las máquinas síncronas [63].

El modelo de bus de compensación distribuido está basado en el concepto del centro de potencia y este consiste en la distribución de pérdidas entre todos los generadores[15].

Este modelo se obtiene reescribiendo las ecuaciones de potencia activa del bus de compensación y los buses PV de la siguiente forma:

$$P_G = (1 + k_G \Upsilon) \qquad 1.3$$

Dónde:

P_G es la potencia generada que se convierte
k_G es una constante de pérdidas de los generadores
Υ es el factor de participación de los generadores en las pérdidas.

Continuación de flujo de carga

Este algoritmo CPF es una innovación para el análisis de sistemas de potencia. Consiste de un paso de predicción que calcula un vector tangente y un paso de corrección que tienen por objeto la parametrización de las ecuaciones no lineales como la ecuación 1.4

$$\begin{bmatrix}0\\0\end{bmatrix} = \begin{bmatrix}(\lambda + \Upsilon k_G)P_{Go}\\ \lambda(P_{Lo}, Q_{Lo})\end{bmatrix} \qquad 1.5$$

Flujo de carga óptimo

El OPF es definido como un algoritmo de optimización con un Método de Punto Interior (IPM) y un método predicción-corrección para problemas de mercado, utilizando variables de holgura para maximizar los beneficios del costo por MWh y limitantes como las constantes térmicas de las líneas de transmisión, límites de generación y seguridad [38].

PSAT permite calcular y graficar los resultados obtenidos del análisis de flujo de carga a partir de la matriz de variables de estado y la matriz Jacobiana, pero calcular estos resultados puede ser problemático si el sistema es de gran escala. Por esta razón, el análisis de pequeña señal de PSAT efectúa cálculos dando uso a las propiedades de las matrices dispersas, reduciendo el número de cálculos y optimizando resultados [47].

Dominio de tiempo

Para esta simulación se maneja el método de regresión de Euler y el de Regla Trapezoidal, que son métodos integrales numéricamente estables para resolver ecuaciones diferenciales y algebraicas por separado. Además, el análisis de estabilidad que incluye transitorios, fallas y operación de interruptores, son manejados por funciones integradas que permiten manejar parámetros después de hacer el análisis de flujo de carga [45].

3.5.4 Uso de PSAT para el análisis de Sistemas Eléctricos de Potencia

PSAT: Sus siglas en inglés corresponden a Power System Analysis Toolbox, que en español significa “Caja de Herramientas para el Análisis de Sistemas de Potencia” [33].

PSAT posee una gran diversidad de modelos entre los cuales se tiene flujo de carga, flujo continuo de carga análisis de pequeña señal, simulación en el dominio del tiempo, unidades de medida fasorial, modelos definidos por el usuario, modelos para turbinas de

viento (aerogeneradores), filtros de conversión de datos de varios tipos de programas comerciales, interfaces con otros programas como el GAMS y el UWFLOW, exportar resultados al formato EPS, texto plano, MS-Excel y Latex.

PSAT posee librerías de uso amigable para el diseño gráfico de redes y todo esto soportado por una interfaz gráfica para el usuario y el uso en líneas de comandos que lo hace completamente flexible y adaptable a otros entornos de programación sobre Matlab. Además, es compatible con el programa tipo GNU Octave lo cual lo hace una herramienta completamente libre de uso extendido [53].

System Power Analysis Toolbox (PSAT), es un OSS para análisis de sistemas de potencia diseñado como soporte para Matlab y GNU Octave. Simula flujos de potencia (PF), continuación de flujos de potencia (CPF), flujos óptimos de potencia (OPF), análisis de estabilidad de voltaje (CPF-VS) y de estabilidad de señal pequeña (SSA), además dominio de tiempo (TD) incluyendo modelos de cargas no convencionales, máquinas síncronas y asíncronas, reguladores y FACTS, todo integrado en una GUI con un editor de diagramas unifilares basado en Simulink del que pueden convertirse estos diagramas unifilares a archivos de matrices para utilizarlos en una consola de programación [48].

PSAT fue diseñado originalmente como herramienta para Matlab, sin embargo, su característica especial de OSS permite que GNU Octave sea la plataforma que concluya la apertura de su código.

El algoritmo de flujo de potencia es el Kernel de PSAT, además este algoritmo se encarga de reiniciar las variables de estado del sistema para obtener las soluciones del flujo. PSAT se beneficia de las cualidades matriciales de Matlab y GNU Octave utilizando matrices dispersas para optimizar resultados, además aprovecha su compatibilidad con otros simuladores basados en Matlab para extender sus habilidades (por ejemplo, contiene interfaces para UWPFLOW y GAMS para resolver CPF y OPF, respectivamente). A continuación se presenta la Figura 3.16 que refleja la arquitectura de PSAT, la Tabla I una comparación de las plataformas que permiten su ejecución y la separación de sus modelos de simulación

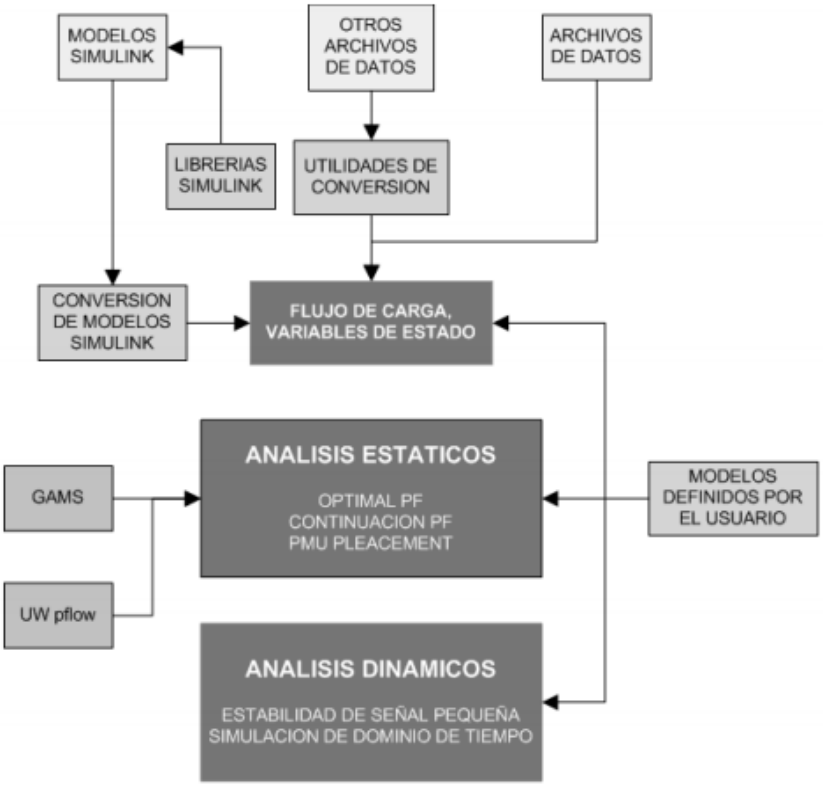

Fig.3.16 ***Esquema sinóptico de PSAT*** **[58].**

Función	Matlab	Octave
CPF	✓	✓
OPF	✓	✓
SSA	✓	✓
TD	✓	✓
Librería Simulink GUI	✓	X
Conversión de formatos de datos	✓	✓
Usuarios definidos	✓	X
Comando de línea	✓	✓

Tabla II ***Comparación del soporte de Matlab y Octave con PSAT***

CAPITULO IV

FLUJO ÓPTIMO DE POTENCIA

4.1 Formulación.

Dado un sistema de 4 nodos y 4 líneas, como se muestra en la figura, solucionar con el método de Newton Raphson.

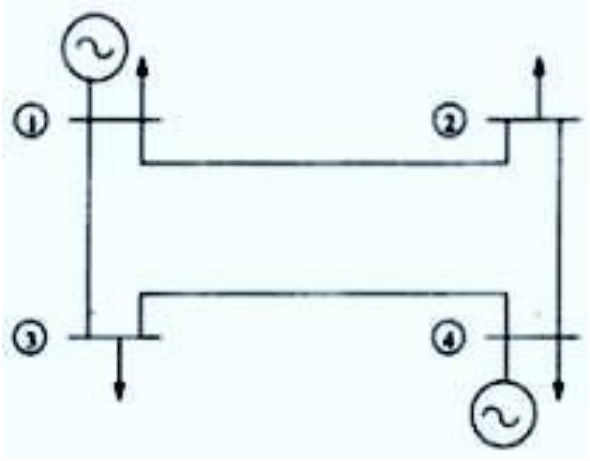

Fig. 4.1 ***Red Eléctrica de 4 nodos***

Datos de línea

Línea	Nodo Inicial y Final	Z serie en p.u.	Y sh/2 en p.u
L1	1-2	0.01008 + j0.05040	0.05125
L2	1-3	0.00744 + j0.03720	0.03875
L3	2-4	0.00744 + j0.03720	0.03875
L4	3-5	0.01272 + j0.06360	0.06375

Datos de barra

Barra	Ps	Qs	PA	QA	V	TIPO DE NODO
	p.u.	p.u.	p.u.	p.u.	p.u.	
1	-	-	0.50	0.3099	1.0 < 0°	Slack
2	0	0	1.70	1.0535	– –	Carga
3	0	0	2.00	1.2394	– –	Carga
4	3.18	-	0.80	0.4958	1.02 -	Generador

Con la aplicación del método de Newton Rapson

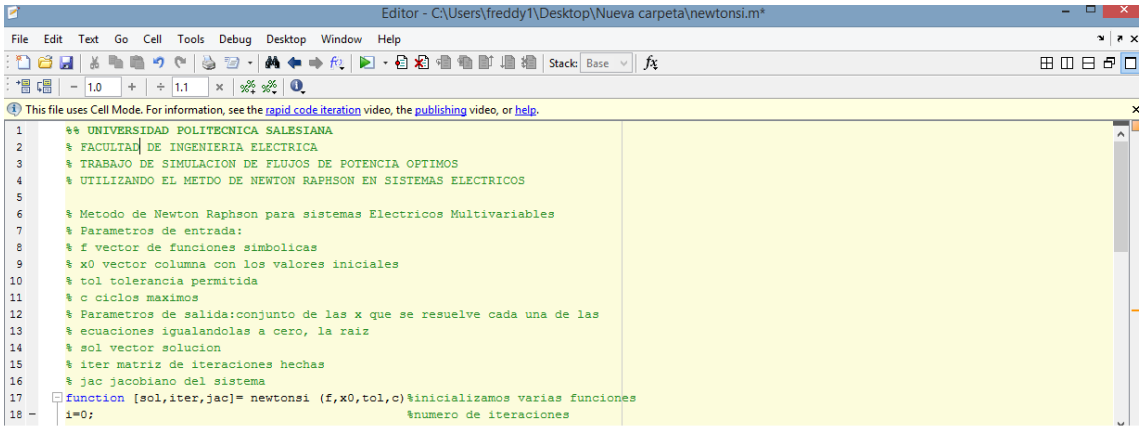

Fig. 4.2 ***Código de programación Matlab***

bus_i	Type	Pg	Qg	Pd	Qd
1	1	0	0	0.5	0.3099
2	3	0	0	1.7	1.0535
3	3	0	0	2	1.2394
4	2	3.18	0	0.8	0.4958

Bsh	Vm	Va	Base KV	Vmax	Vmin
0	1	0	230	1.05	0.95
0	1	0	230	1.05	0.95
0	1	0	230	1.05	0.95
0	1.02	0	230	1.05	0.95

N.inicio	N.final	R	X	Bsh/2
1	2	0.01008	0.0504	0.05125
1	3	0.00744	0.0372	0.03875
2	4	0.00744	0.0372	0.03875
3	4	0.01272	0.0636	0.06375

En la aplicación del sistema propuesto y al desarrollar las matrices con los datos asignados, se deducen un sistema de ecuaciones no lineales de manera simultánea que en muchos problemas de ingeniería requieren la solución de un conjunto de ecuaciones y para ello el método de Newton Raphson se puede extender para resolverlas.

$$f_1(x_1, x_2, x_3 \dots\dots, x_n) = 0$$
$$f_2(x_1, x_2, x_3 \dots\dots, x_n) = 0$$
$$f_n(x_1, x_2, x_3 \dots\dots, x_n) = 0$$

Si las funciones $f_1, f_2, \dots., f_n$ se expanden en la serie de Taylor alrededor de un punto solución arbitraria y si solo se consideran los términos lineales, obtener los valores de las funciones en la solución exacta como

$$f_1(x_1^*, x_2^*, x_3^* \dots\dots, x_n^*)$$
$$\approx f_2(x_1, x_2, x_3 \dots\dots, x_n) + \left(\frac{\delta f_1}{\delta x_1}\right)(x_1^* - x_1)$$
$$+ \left(\frac{\delta f_1}{\delta x_2}\right)(x_2^* - x_2) \dots., + \left(\frac{\delta f_1}{\delta x_n}\right)(x_n^* - x_n)$$

Donde las derivadas parciales son evaluadas en $(x_1, x_2, x_3 \dots\dots, x_n)$ haciendo que todas las ecuaciones se igualen a cero y expresando en el sistema en forma matricial tenemos

$$0 = \begin{pmatrix} f_1 \\ f_2 \\ \dots \\ f_n \end{pmatrix} + \begin{pmatrix} \frac{\delta f_1}{\delta x_1} & \frac{\delta f_1}{\delta x_1} & \frac{\delta f_1}{\delta x_1} \\ \frac{\delta f_1}{\delta x_1} & \frac{\delta f_1}{\delta x_1} & \frac{\delta f_1}{\delta x_1} \\ \dots\cdot & \dots\cdot & \dots\cdot \\ & \frac{\delta f_1}{\delta x_1} & \end{pmatrix} \begin{pmatrix} x_1^* - x_1 \\ x_2^* - x_2 \\ \dots\dots. \\ x_n^* - x_n \end{pmatrix}$$

Donde la matriz de las derivadas parciales de f_1 es la matriz Jacobiana $J_f\,(x)$ de aquí se obtiene la ecuación [35].

$$x^{(k-1)} = x^{(k)} - \frac{f\,(x^k)}{J_f\,(x^k)}$$

O de la otra forma

$$x^{(k-1)} = x^{(k)} - J_f^{-1}(x^{(k)})(x^{(k)})$$

Para hacer los cálculos del método de Newton, no se requiere invertir la matriz Jacobiana en casa iteración.

En nuestro caso el código de programación está definido de la siguiente manera.

```
function [sol,iter,jac]= newtonsi (f,x0,tol,ci=0;
jac = jacobian (f);
vars = findsym(f);
deltaX = x0;
iter= [x0' 0];
while norm (deltaX)> norm(x0)*tol && i<c
    fx0 = subs (f,vars,x0);
    dfx0 = subs (jac,vars,x0);
    deltaX = dfx0\(-fx0);
    x0 = x0+deltaX;
    i=i+1
    iter=[iter;x0' norm(deltaX)];

end
if i<c
    sol = x0;
else
    sol = 'No converge';
end
```

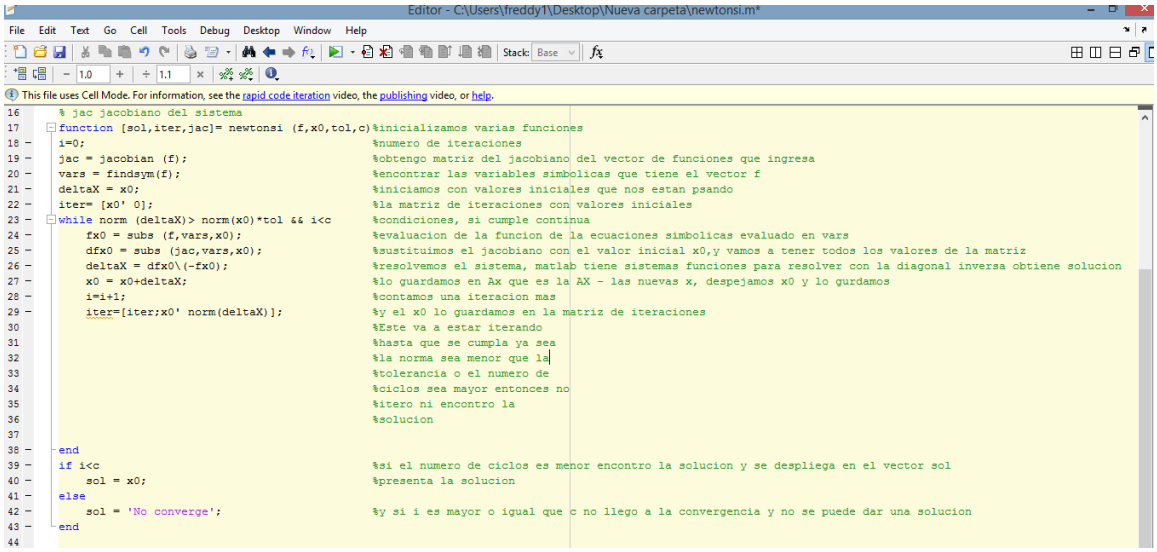

Fig. 4.3 ***Código de programación Matlab***

Ahora para nuestro caso del ejemplo vamos a Matlab y construimos nuestra función el código anterior lo guardamos con el nombre newtonsi el programa guarda el archivo con la extensión .m

Primero creamos las variables simbólicas y a continuación creamos el vector de las funciones no líneas que le llamamos A

syms x1 x2 x3 para el sistema nuestro tenemos 3 ecuaciones no lineales que son;

A=[x1^2+2*x2^2+exp(x1+x2)+x1*x3-6.1718

10*x2+x2*x3

sin(x1*x3)+x2^2+x1-1.141]

Luego hacemos la llamada de la programación del archivo newtonsi.m, inicializando

todo esto con 1, y luego corremos nuestro programa

```
>> syms x1 x2 x3
>> A=[x1^2+2*x2^2+exp(x1+x2)+x1*x3-6.1718
10*x2+x2*x3
sin(x1*x3)+x2^2+x1-1.141]
 A =
  exp(x1 + x2) + x1*x3 + x1^2 + 2*x2^2 - 30859/5000
                              10*x2 + x2*x3
             x2^2 + x1 + sin(x1*x3) - 1141/1000
 >> x0=[1;1;1]
x0 =
     1
     1
     1
```

En el momento que inicializamos con 1 y llamando al programa newton.si se despliega el vector solución la matriz con las interaciones y el jacobiano. Se pasa con los parámetros el vector de las funciones no lineales construidas con funciones simbólicas y el vector de los valores iniciales, la tolerancia permitida y el número máximo de ciclos. Ejecutando tenemos

>> [sol,iter,jac]=newtonsi(A,x0,0.00001,100)

```
sol =
    1.4695
    0.0000
   -0.2278
iter =
    1.0000    1.0000    1.0000         0
    1.5883    0.1538   -0.6922    1.9814
    1.4401   -0.0131    0.1023    0.8253
    1.4704   -0.0004   -0.2340    0.3378
    1.4695    0.0000   -0.2278    0.0063
    1.4695    0.0000   -0.2278    0.0000
jac =
[ 2*x1 + x3 + exp(x1 + x2), 4*x2 + exp(x1 + x2),            x1]
[                        0,            x3 + 10,            x2]
[       x3*cos(x1*x3) + 1,               2*x2, x1*cos(x1*x3)]
```

Utilizando el GUI para el desarrollo del mismo programa se tiene de la siguiente manera

Se programa dependiendo la necesidad de las aplicaciones, para ello con las opciones que nos brinda el GUI podemos crear opciones interactivas para proceder a programar en cada uno de ellos.

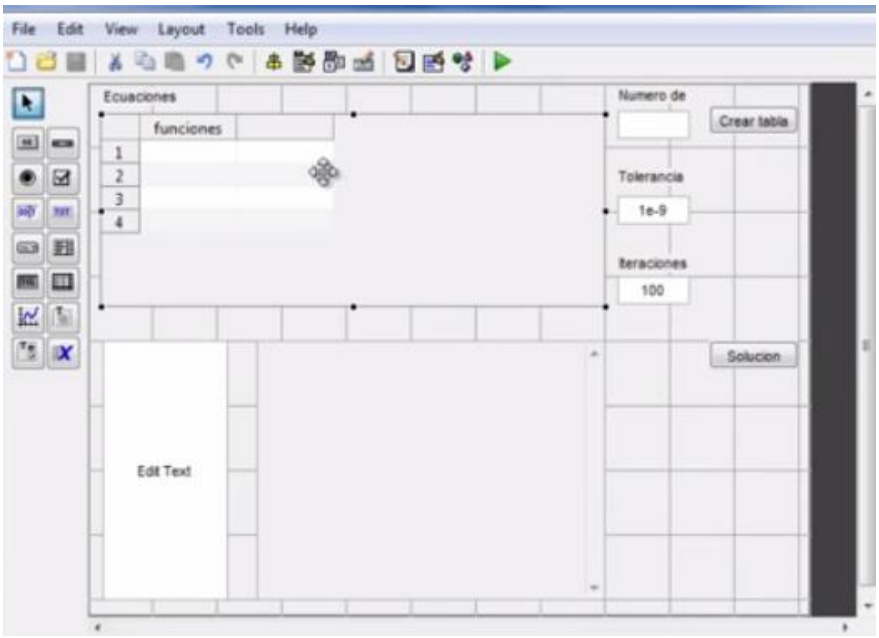

Fig. 4.4 ***Ventana principal para la programación***

En cada uno de los aplicativos se configura de la siguiente manera

Para la creación de Gui Table donde se colocaran las ecuaciones se configura así

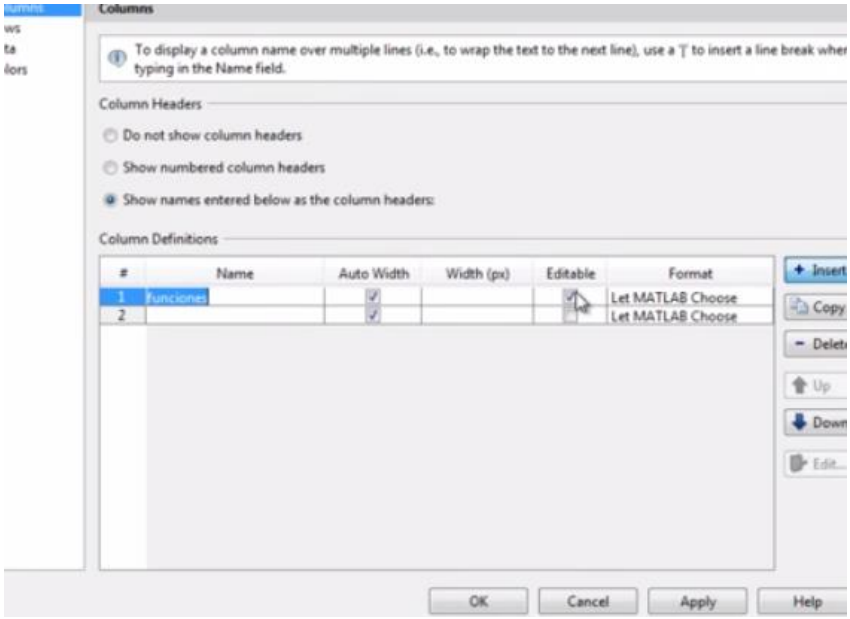

Fig. 4.5 ***Ventana principal para la programación***

En la figura 4.5 cambiamos en el columname indicamos el nombre para nuestro caso es funciones, se habilita el listón editable y se elimina el ultimo renglón para que tenga 2 columnas la tabla. El tag se le denomina table.

En el caso del siguiente elemento es un Edidtex, las propiedades que se cambia max igual a 100 que nos permite teclear 100 veces máximo el tag colocamos es inicial.

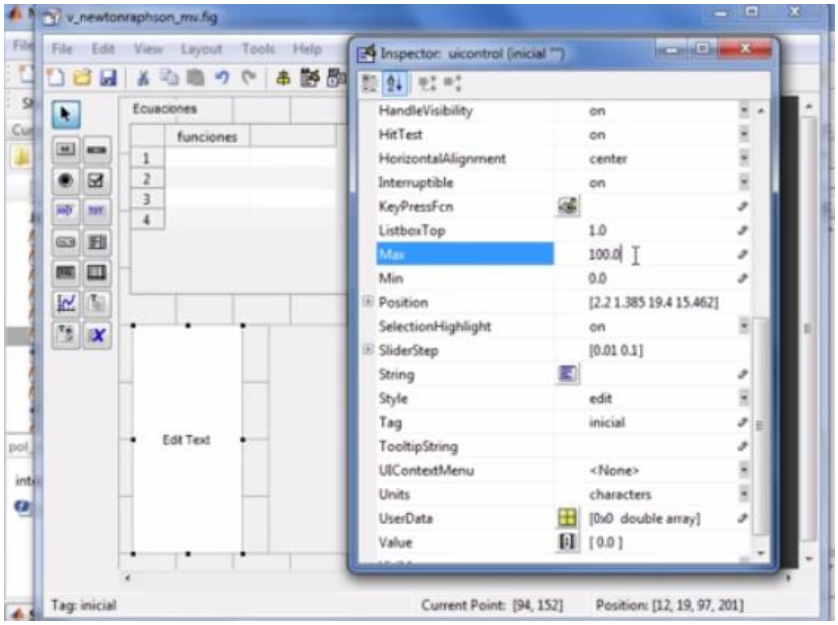

Fig. 4.6 ***Ventana editable cambiando propiedades***

Para el caso del otro cajón Edittex se lo cambia la propiedad max a 100 donde se muestra todos los valores del vector solución, como muestra la solución no debe ser editable y por ello se la deshabilita.

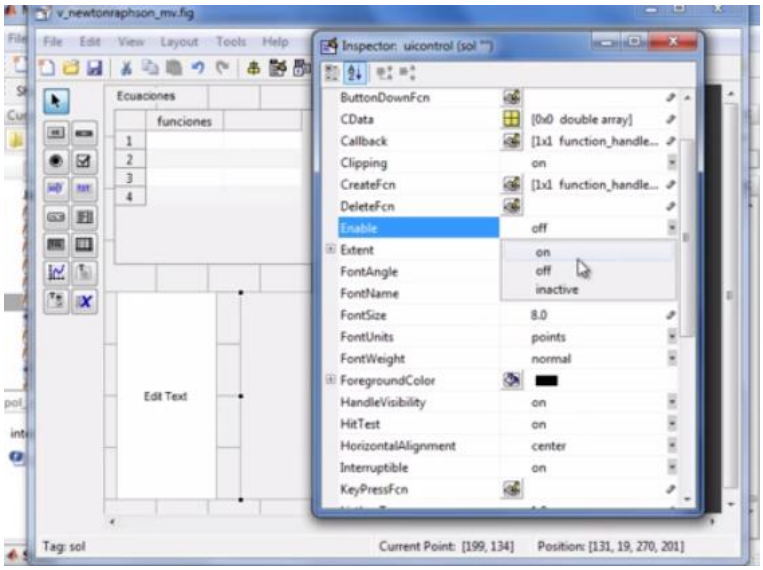

Fig. 4.7 ***Ventana editable donde se despliega la solución***

Otras Edittex en el caso del botón denominado Numero de ecuaciones, es importante hacerlo de forma dinámica puesto que no sabemos el número de ecuaciones que tendrá el sistema necesitamos solicitar ese número, el tag se denomina n.

La tolerancia es un Edittex, aquí lo que se hace es que se coloca la tolerancia definida como defaul 1*10^9 el tag es tol. También el número de iteraciones queda con 100 en el siguiente EditText.

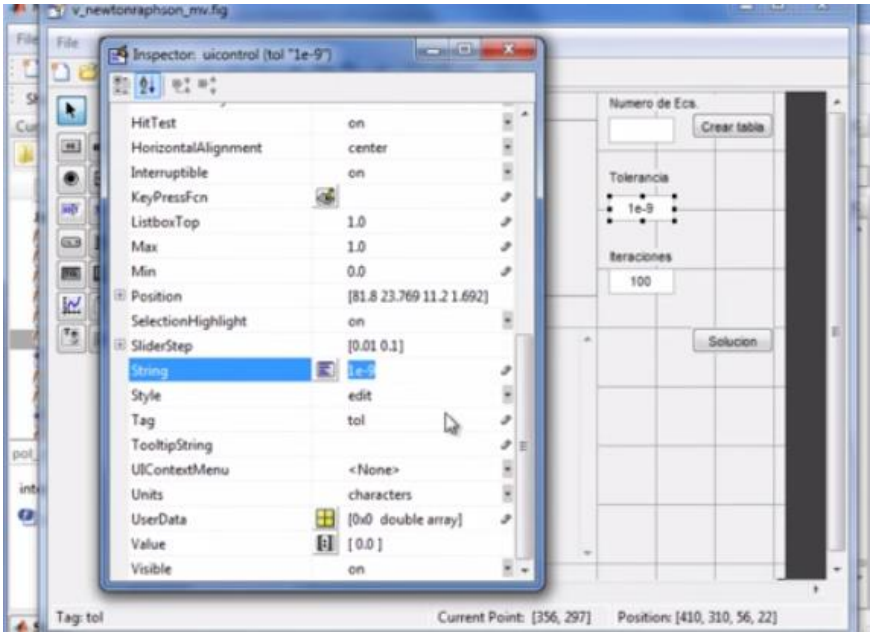

Fig. 4.8 ***Ventana de iteraciones definida***

Por ultimo coloque los dos botones Push Button al primero se define crear la tabla que va a generar el número de renglones dependiendo el valor de ecuaciones que solicite y el otro que genera la solución para la optimización del sistema.
El código de programación de Simulink es el siguiente para el botón solucion:

```
% --- Executes on button press in solucion.
function solucion_Callback(hObject, eventdata, handles)
% hObject    handle to solucion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
A=cell(get(handles.tabla,'data'));
tol=str2double(get(handles.tol,'string'));
iter=str2double(get(handles.iter,'string'));
x0=str2num(get(handles.inicial,'string'));
M=sym(A);
[sol]=newtonsi(M,x0,tol,iter);
set(handles.sol,'string',num2str(sol));
```

Se crea como una celda todos los valores de la tabla que aquí ya tenemos calculados las funciones no lineales de todas las funciones en todos los renglones que crea el renglón tol.
Se crea una matriz con los valores que se leyeron con los valores de las ecuaciones no lineales y creamos el vector simbólico de todas esas funciones y se visualiza dependiendo las iteraciones colocando la solución en la variable sol y cambiando a text se demuestra cómo sigue

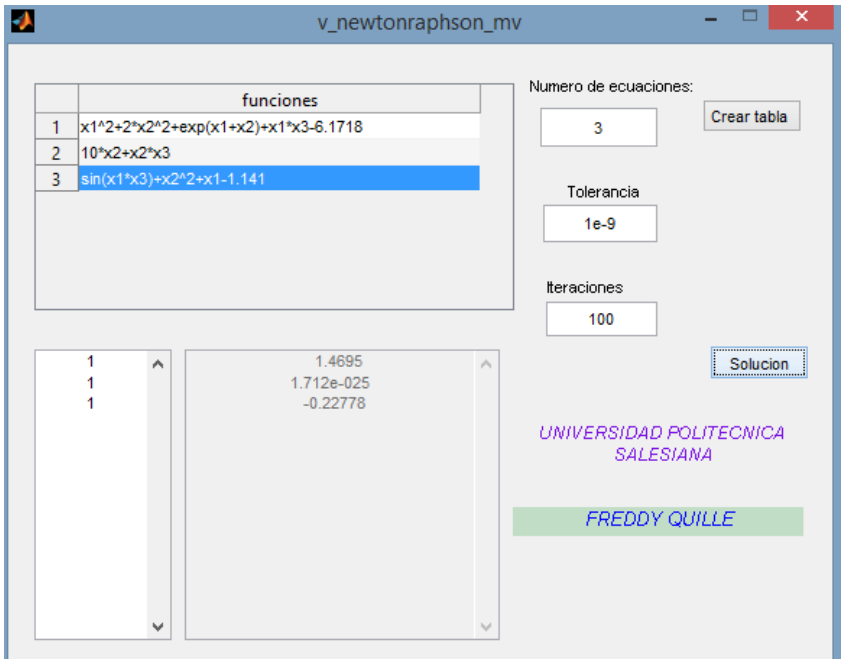

Fig. 4.9 ***Ventana donde se verifica la solución óptima del sistema***

4.2 Casos de estudio

POTENCIA REACTIVA CON FLUJO ÓPTIMO POTENCIA.

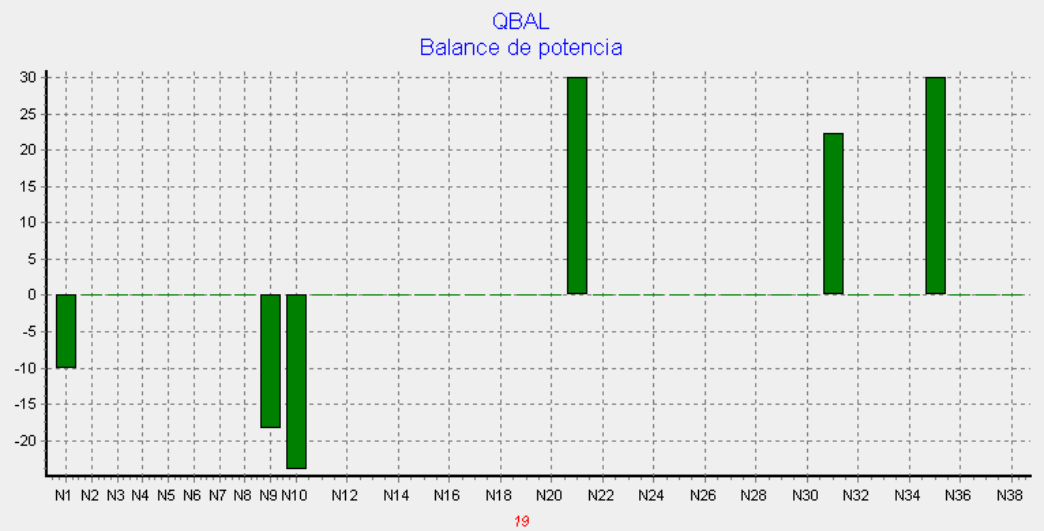

Figura 4.9 ***Potencia Reactiva con Flujo Óptimo Potencia.***

En la gráfica 4.9 se representa el valor potencia reactiva con flujo óptimo de potencia en MVAR. Con relación a cada uno de los nodos que se obtiene en la distribución del Sistema Nacional Interconectado el análisis se lo hace por cada hora de trabajo independientemente en un rango de tiempo [52]. Se puede observar como aumenta la potencia de salida de las unidades generadoras con el aumento de la demanda.

POTENCIA ACTIVA CON FLUJO ÓPTIMO DE POTENCIA.

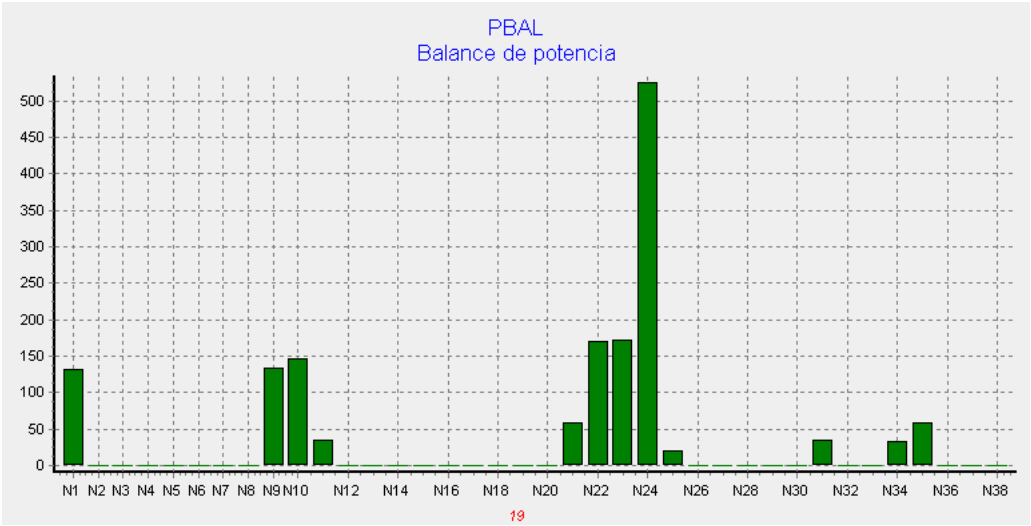

Figura 4.10 ***Potencia Activa con Flujo Óptimo Potencia***

En la gráfica 4.10 se representa el valor potencia activa con flujo óptimo de potencia en M.W. con relación a cada uno de los nodos que se obtiene en la distribución del Sistema Nacional Interconectado el análisis se lo hace por cada hora de trabajo independientemente en un rango tiempo. Se puede observar como aumenta la potencia de salida de las unidades generadoras con el aumento de la demanda.

POTENCIA ACTIVA CON FLUJO NORMAL.

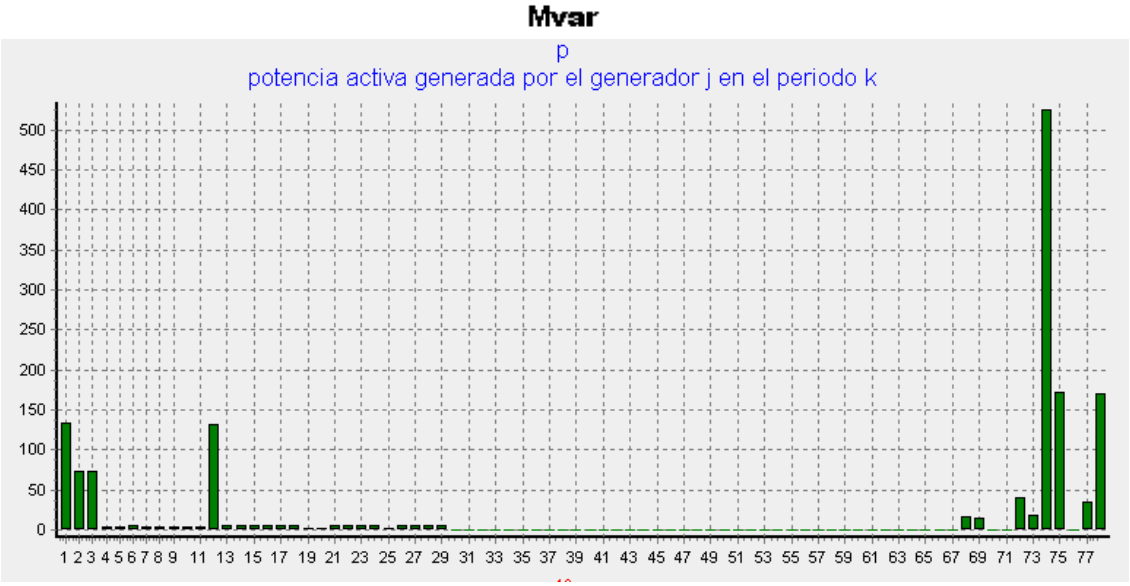

Figura 4.11 ***Potencia Activa con Flujo. Normal***

En la gráfica 4.11 se representa el valor de potencia activa en MW que entrega cada una de las unidades generadoras en una hora especifica el análisis se lo hace independientemente en tiempo[55]. Se puede observar claramente que con el aumento de la demanda se aumenta la potencia de salida.

POTENCIA REACTIVA CON FLUJO NORMAL

Figura4.12 ***Potencia Reactiva con Flujo Normal***

La grafica 4.12 representa la potencia reactiva en MVAR que entrega cada una de las unidades generadoras en una hora específica, el análisis se lo hace en un rango de tiempo de 1 a 25 horas en nuestro caso de estudio se lo hizo a la hora 7. Se puede observar como aumenta la potencia de salida de las unidades generadoras con el aumento de la demanda.

4.3 Análisis de resultados

Se resolvió las ecuaciones del sistema de potencia con la ayuda del método de Newton Raphson aplicando la programación no lineal, del flujo óptimo con restricciones de balance de potencia, como resultado de este análisis los niveles de voltaje subieron representativamente al optimizar el flujo óptimo en relación con el flujo normal, esto es a la redistribución de potencia reactiva que a su genera la disminución de las pérdidas de potencia activa del sistema.

En el área de ingeniería eléctrica, la determinación de las condiciones de operación en estado permanente de un sistema de potencia se realiza por medio de un estudio de flujos de potencia, el cual es no lineal por naturaleza por lo que requiere del cálculo del Jacobiano que se evaluará por medio del Newton Raphson.

El sistema de ecuaciones no lineales de potencias formado para la solución del problema de flujos de potencia está constituido se resuelve con el algoritmo realizado para esta aplicación.

Se resuelve el Jacobiano para un sistema de 4 nodos tomado de la referencia, mostrando resultados para la tercera iteración la cual es en la que converge el método de Newton Raphson. Los resultados obtenidos para el cálculo del Jacobiano usando ambos métodos, esto es, utilizando las expresiones de la programación presentada con la aplicación de Matlab, evalúa el método que proporcionan el Jacobiano exacto, y utilizando diferenciación automática.

Se observa que los resultados son exactamente los mismos al ejecutar el script así como en la simulación utilizando un GUI aplicando el Simulink, los valores son idénticos ya que las operaciones numéricas que se realizan son hasta el final en ambas formulaciones, lo cual indica que no hay error por truncamientos, por lo tanto la precisión es la que tenga la computadora al ejecutar el algoritmo planteado en MATLAB.

CONCLUSIONES Y RECOMENDACIONES.

Se concluye que para la demostración del procedimiento de la diferenciación automática es eficiente y concluyente para el cálculo de derivadas exactas, gracias a la implementación con la herramienta del programa computacional MATLAB aplicando del método de Newton Raphson en la solución del problema de flujos de potencia. Su solución fue ejecutada forma tradicional puesto que nos entrega el Jacobiano exacto, y con ello proceder a compararlo con la solución obtenida por diferenciación automática presentado por el algoritmo desarrollado. Ya que los resultados son iguales, se concluye que el cálculo de derivadas por medio de diferenciación automática es exacto, por lo que esta técnica resulta eficiente cuando se requiere del cálculo de derivadas, en este caso en la forma de vector Jacobiano de una función planteada.

Se desarrolló el programa de análisis de flujo de carga pudiéndose analizar SEP de gran tamaño con el ingreso de datos preestablecidos en la ejecución del algoritmo. Se desarrolló una aplicación para la optimización de flujos de potencia en modelado AC, con la limitante del análisis de un sistema en específico debido a la complejidad de este tipo de modelaje que toma en cuenta la forma de inyección de la potencia real y reactiva de manera sinusoidal y su complejidad de programación.

Las simplificaciones efectuadas al modelo AC de OPF, dieron lugar al modelaje DC, se tomó como metodología de solución el método iterativo de Newton por ser este uno de los métodos más poderosos para resolver este tipo de problemas no lineales y uno de los más conocidos y práctico para el entorno de Matlab.
Se recomienda la utilización de programas poderosos como es el Matlab con la preparación adecuada y así seguir ahondando en el presente tema de investigación de Sistemas de Potencia.
Así como también investigar y crear métodos matemáticos alternativos, para una convergencia de los sistemas eléctricos se brinde algoritmos que satisfagan las necesidades más prioritarias y eviten las pérdidas de potencia en las líneas de trasmisión.

Analizar e investigar modelos matemáticos alternativos, para la solución más rápida y eficaz con la ayuda del modelamiento matemático con la ayuda de sistemas matemáticos. Elaborar el modelaje de una optimización de flujo de potencia para análisis de manera general e incluir las restricciones técnicas de voltaje, ángulo, y límites en la inyección de potencia de los generadores.

La optimización del flujo óptimo de potencia es un modelamiento altamente importante para poder lograr la programación de la operación con un nivel de seguridad aceptable mientras se optimiza una función objetivo, que puede constituir entre otros aspectos más importantes los costos de producción o pérdidas de transmisión.

Las aplicaciones para el flujo óptimo de potencia (activa o reactiva) como se muestra en esta investigación son aplicables tanto para el Sistema Eléctrico de Potencia, así como para especificar diferentes políticas operativas, las limitaciones en los equipos y requerimientos de seguridad, y analizar las diferentes implicaciones.

Gracias a la optimización del flujo de potencia, serviría como una metodología para poder retomar un nivel de operación seguro. Es decir, en un caso real, el sistema pasará a un estado de operación más seguro, en el caso que se registre una perturbación alguna falla y los elementos del sistema se sobrecarguen o exista una violación en el nivel de voltaje.

Con el mismo modelo de optimización planteado en este trabajo se puede incluir sin mayor complejidad, restricciones para modelar los taps de los transformadores, modelar los bancos de capacitores y cualquier otra restricción que se requiera analizar.

Para sistemas más complejos donde se tiene prioritariamente que utilizar software ampliamente amigable con el entorno de solución se recomienda la utilización del método Newton-Raphson completo. Como se trata de ganar al tiempo plantearían algoritmos de solución considerando el algoritmo desacoplado rápido para el flujo de potencia.

Gracias a la utilización de algoritmos se hace más confiable y se da soluciones que son altamente efectivas por ejemplo se puede crear un mercado de potencia reactiva, paralelo al de potencia activa, donde los generadores participantes ofrezcan un precio por su servicio, ante esta situación, se puede recurrir a la programación no lineal para resolver el despacho de potencia reactiva, para ello tendría que reformarse también el despacho económico de potencia activa, ya que la variable costo por potencia reactiva sería otra restricción en su proceso de programación, la solución propuesta en este caso sería realizar un solo procedimiento para determinar el despacho de potencia activa y reactiva utilizando la programación no lineal, la función objetivo sería el costo operativo y entre sus restricciones, las ecuaciones de flujo de potencia.

Con la aparición de Matlab y otros programas de simulación se hace efectiva la resolución de casos de estudio aplicados a flujos de potencia.
Permitiendo el desarrollo de las aplicaciones para la resolución efectiva de sistemas de potencia, para ello se requiere el conocimiento previo del tema previo la utilización de esta herramienta.

Con lo que se refiere al proyecto desarrollado en esta tesis, se ha utilizado el PSAT la cual se convierte en una opción factible para muchos usuarios, gracias a su amplia versatilidad y poderosa herramienta de MATLAB. Se puede mencionar que es bastante confiable para la utilización, la utilidad que presenta la facilidad de interactuar con programas adicionales facilitando las operaciones simétricas, en el campo económico, gracias a las simulaciones efectuada.

Se recomienda ampliar este programa con más restricciones ya que las que se han propuesto para la selección de unidades y flujo óptimo no son las únicas, si se desea agregar más restricciones al U.C. Se lo puede hacer variando el código fuente del archivo que usa el optimizador para la solución de los problemas antes mencionados.

Referencias:

[1] G. A. Rios, *Análisis y Control de Sistemas Electricos de Potencia*. Quito, 1988, p. 95.

[2] F. M. Gonzalez Longatt, “Introducción a los Sistemas de Potencia,” p. 57, 2008.

[3] R. Donald and A. Caisse, “Máquinas Rotativas y Transformadores,” 1997.

[4] R. Jiménez, “Simbología y Normas Eléctricas,” pp. 1–48.

[5] C. P. Ibañez, “Análisis del Motor de Inducción Mediante Técnicas de Programación no Lineal,” p. 56, 2010.

[6] A. Novales, “Análisis de Regresión.” 2010.

[7] M. Merino, “Técnicas Clásicas de Optimización,” .

[8] S. Estrada, “Programacion No Lineal.”

[9] A. Medrano, *Optimización para Ingenieros con Herramientas Informáticas*. 2014.

[10] N. del L. S. F. Scenna, *Modelado, simulación y optimización de procesos químicos*. 1999, p. 840.

[11] I. Gr, “Apuntes Matlab CCD,” .

[12] F. Guerrero, “Diseño de Interfaz Gráfica en Matlab,” 2014, pp. 56–74.

[13] A. Bustos, M. Escobar, and H. Rubio, “Modelo Paramétrico de la Mecánica de Robot Mimbot,” p. 334, 2010.

[14] D. (Universidad V. Guzman, “Flujos de Potencia con Matlab,” p. 110, 2012.

[15] R. P. C. Steven C.Chapra, *Métodos Numéricos Para Ingenieros*, Quinta Edi. p. 977.

[16] Kennedy J. Yeberhart R. C., “Particle Swarm Optimization”, Proceedings of the IEEE International Conference on Neural Networks, 1995.

[17] Ting T. Yrao M., “A novel approach for unit commitment problem via an effective hybrid particle swarm optimization”, IEEE Trans. on Power System, 2006

[18] L. Gyugyi, T. R. Rietman, A. Edris, C. D. Schauder, D. R. Torgerson, S. L. Williams, "The Unified Power Flow Controller a New Approach to Power Transmission Control," IEEE-Trans, on Power Delivery, Vol. 10, No. 2, April, 1995, pp. 1085-109

[19] Toscano G. Y Coello C., “A constrainthandling mechanism for particle swarm optimization”, Proceedings of the 2004 Congress on Evolutionary Computation, IEEE, 2004.

[20] Wood A. y Wollenberg B., “Power Generation: Operation and Control”, Ed. Wiley, Nueva York, 1996.

[21] Mirko,T&Dragoslav.R (2006), 'An initialization procedure in solveing optional power flow by algoritm genet',IEEE Transactions on power systems (02),1.

[22] Momoh,J(1989),'Optimal power flow whit multiple objective funtions',IEEE Dpto.of Electrical Engieering Howar University.

[23] F. D. Galiana, K. Almeida, M. Toussaint, J. Gdriffin, D. Atanackovic, B. T. Ooi, D. T. McGillis, "Assessment and control of the impact of FACTS devices on power system performance," IEEE- Trans, on Power Systems, Vol. 11, No. 4, November, 1996, pp. 1931- 1936.

[24] M. Huneault, F. D. Galiana, "A Survey of the Optimal Power Flow Literature," IEEETrans. on Power Systems, Vol. 6. No. 2, May, 1991, pp. 762-770

[25] G. Gutiérrez A., Método de Newton para Resolver Problemas de Flujos Optimos: Aplicación a la Evaluación de Acceso a la Transmisión en Transacciones a Corto Plazo, Tesis de Maestría, Instituto Tecnológico de Morelia, Diciembre, 1996.

[26] G. Alcaraz, M. M. Martinez, J. H. T. Hernández, "Comparison of Alternatives for the Inclusion of Real Power Flow Constraints in Optimal Power Flow by Newton's Method," 28th North American Power Symposium, Boston, Mass., November, 1997

[27] L. Garver. “Transmission Network Estimation Using Linear Programming”. IEEE Transactions on Power Apparatus and Systems, Vol. 89, Sep./Oct. 1970. pp. 1688-1697.

[28] R.Villasana, L. L. Garver and S. J. Salon. “Transmission Network Planning Using Linear Programming”. IEEE Transactions on Power Apparatus and Systems. Vol. PAS-104, No. 1, Feb. 1985, pp. 349–356.

[29] I. G. Sánchez, R. Romero, J. R. S. Mantovani and M. J. Rider. “Transmission–Expansion Planning Using the DC Model and Nonlinear–Programming Technique”. IEE Proceedings of Generation, Transmission and Distribution. Vol. 152, No. 6, Nov. 2005, pp. 763–769.

[30] M. Rider, L.A. Gallego, R. Romero and A. García. “Heuristic Algorithm to Solve the Short Term Transmission Network Expansion Planning”. Conference Record, Industry Applications Society, IEEE-IAS Annual Meeting, Vol. 2007, p. 1-8, 2007

[31] M. Dorigo, V. Maniezzo and A. Colorni. “The Ant System: Optimization by a colony of cooperating agents”. IEEE Transactions on Systems, Man, and Cybernetics–Part B, Vol. 26, No.1, 1996, pp. 1-13.

[32] Milan Bjelogrlic, Milan S. Calo vie, Borivoje S. Babic., "Application of Newton's Optimal Power Flow in Voltage/Reactive Power Control," IEEE Trans, on Power Systems, Vol. 5, No. 4, November 1990, pp. 1447-1453.

[33] M. Madrigal, L. Coria "Uso de Asignación Dinàmica de Memoria para el Manejo y Solución de Sistemas de Ecuaciones Lineales Dispersas," Memorias RVP'96, IEEE Sección México, Julio, 1996., Acapulco Grò

[34] M. Dorigo and L. Gambardella. “Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem”. IEEE Transactions on Evolutionary Computation. Vol. 1 Nº 1, 1997b, pp. 53-66.

[35] Jerosolimsky M., and Levacher L. Mayo 1994. A New Method for Fast Calculation of Jacobian Matrices: Automatic Differentiation for Power System Simulation, IEEE Transactions on Power Systems, Vol. 9, No. 2, pp. 700-706.

[36] R. Divi, H. K. Kesavan, "A Shifted Penalty Function Approach for Optimal Load-Flow," IEEE-Trans, on Power Systems, Vol. PAS-101, No. 9, September, 1982, pp. 3502-3512

[37] L. Gyugyi, "Dynamic Compensation of AC Transmission Lines by Solid-State Synchronous Voltage Sources", Presented at IEEE 1993 Summer Meeting, Paper No. 93 SM 434-1 PWRD

[38] A. Monticelli, M. V. F. Pereira, S. Granville "Security-Constrained Optimal Power Flow with Post-contingency Corrective Rescheduling," IEEE-Trans, on Power Systems, Vol. PWRS-2, No. 1, February, 1987, pp. 175-182.

[39] Orfanogianni Tina. 2000. A flexible software environment for steady-state power flow optimization with series FACTS devices. Tesis Doctoral, Swiss Federal Institute of Technology (ETH) Zurich.

[40] Ajjarapu V., and Ibsais A. Mayo 1997. The Role of Automatic Differentiation in Power System Analysis, IEEE Transactions on Power Systems, Vol. 12, No. 2, pp. 592- 597.

[41] J. F. Dopazo, 0. A. Klitin, G. W. Stagg, M. Watson, "An Optimization Technique for Real and Reactive Power Allocation," Proc. of the IEEE, Vol. 55,No. 11,1967,pp. 1877-1885.

[42] A. Nabavi-Niaki, M. R. Iravani "Steady-state and Dynamic Models of Unified Power Flow Controller (UPFC) for Power System Studies," IEEE-Trans, on Power Systems, Vol. 11, No. 4, November, 1996, pp. 1937-1949.

[43] H. Chowdhury, Sai fur Rahman, "A Review of Recent Advances in Economic Dispatch," IEEE Trans, on Power Systems, Vol. 5, No. 4, pp. 1248-1259, November, 199

[44] D.I. Sun, T. Hu, G. Lin, C. Lin, C. Chen, "Experiences with Implementing Optimal Power Flow for Reactive Scheduling in the Taiwan Power System,"IEEE/PES Summer Meeting, San Francisco, California, July 12-17, 1987.

[45] R. C. Burchett, H. H. Happ, K. A. Wirgau, 'Large Scale Optimal Power Flow," IEEE Trans, on Power Systems, Vol. PAS-101, No. 10, October, 1982, pp. 3722-3732

[46] M. Noroozian, L. Angquist, M. Ghandhari, G. Andersson, "Use of UPFC for Optimal Power Flow Control," IEEE-Trans, on Power Delivery, Vol. 12, No. 4, October, 1997, pp. 1629-1634.

[47] Averick Brett M., Moré Jorge J., Bishof Christian H., Carle Alan, and Griewank Andreas. 1994. Computing Large Sparse Jacobian Matrices Using Automatic Differentiation. SIAM Journal on Scientific Computing, 15(2):285-294

[48] Coleman T. F., and Verma A., Marzo 1997. ADMIT-1: Automatic Differentiation and MATLAB interface toolbox, Tech. Rep. CTC97TR271, Theory Center, Cornell University.

[49] H. H Happ, "Optimal Power Dispatch - a Comprehensive Survey," IEEE Trans. PAS, Vol. PAS-9 6, No. 3, May/June, 1977, pp. 841-853.

[50] S. Chávez, G. Alcaraz, M. M. Martínez, J. H. T. Hernández, "Flujos Optimos en Coordenadas Rectangulares Resuelto por el Método de Newton," Memorias RVP'97, IEEE Sección México, Julio, 1997, Acapulco Gro.

[51] Katia C. Almeida, Francisco D. Galiana, "Critical Cases in The Optimal Power Flow," IEEE Trans, on Power Systems, Vol. 11, No. 3, August 1996, pp. 1509-1518

[52] A-A. Edris, "Proposed Terms and Definitions for Flexible ac Transmission System (FACTS)," IEEE-Trans, on Power Delivery, Vol. 12, No. 4, October, 1997, pp. 1848- 1853.

[53] D.I. Sun, T. Hu, G. Lin, C. Lin, C. Chen, "Experiences with Implementing Optimal Power Flow for Reactive Scheduling in the Taiwan Power System,"IEEE/PES Summer Meeting, San Francisco, California, July 12-17, 1987

[54] W. F. Tinney, J. W. Walker, "Direct Solutions of Sparse Network Equations by Optimally Ordered Tiangular Factorization", Proceedings of the IEEE, Vol 55, No 11, November, 1967

[55] D. I. Sun, B. Ashley, B. Brewer, A. Hughes, and W. F. Tinney, "Optimal Power Flow by Newton Approach," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-103, No. 10, pp. 2864-2880, October, 1984.

[56] Rall L. B., and Corliss G. F. 1996. An introduction to automatic differentiation. In M. Berz, C. Bischof, G. Corliss, and A. Griewank, editors, Computational Differentiation, Procs Second International Workshop on Computational Differentiation. SIAM.

[57] A. Monticcelli, W. E. Liu, "Adaptative Movement Penalty Method for the Newton Optimal Power Flow," IEEE TPS, Vol. 7, No. 1, pp. 334-342, February, 1992.

[58] H. W. Dommel, W. F. Tinney, "Optimal Power Flow Solutions," IEEE Trans, on Power App. and Systems, Vol. PAS-87, October, 1968, pp. 1866-1876

[59] C. R. Fuerte-Esquivel, E. Acha, "Unified Power Flow Controller: A Critical Comparison of Newton-Raphson UPFC Algorithms in Power Flow Studies", IEE Proc-Gener. Transm. Distrib. Vol. 144, No. 5, September, 1998. pp.437-444.

[60] Gamal A. Maria, J. A. Findlay, "A Newton Optimal Power Flow Program For Ontario Hydro EMS," IEEE Trans, on Power Systems, Vol. PWRS-2, No. 3, pp. 576-584, August, 1987

[61] Sasson A. M., Viloria F., Aboytes F., "Optimal Load Flow Solution Using the Hessian Matrix " IEEE Trans, on Power Systems, Vol. 92, No. l,pp. 31-41, 1973

[62] Ambriz-Pérez, E. Acha, C. R. Fuerte-Esquivel, A. De la Torre, "Incorporation of a UPFC Model in an Optimal Power Flow Using Newton's Method", IEE Proc-Gener. Transm. Distrib. Vol. 145, No. 3, May, 199

[63] R.Mihalic, et.al, "Improvement of Transient Stability Using a Unified Power Flow Controller", Presented at IEEE Winter Meeting, Paper 95 WM 269-1 PWRD

ANEXOS

Código de programacion Matlab desarrollado

```
function varargout = v_newtonraphson_mv(varargin)
% V_NEWTONRAPHSON_MV M-file for v_newtonraphson_mv.fig
%      V_NEWTONRAPHSON_MV, by itself, creates a new V_NEWTONRAPHSON_MV
or raises the existing
%      singleton*.
%
%      H = V_NEWTONRAPHSON_MV returns the handle to a new
V_NEWTONRAPHSON_MV or the handle to
%      the existing singleton*.
%
%      V_NEWTONRAPHSON_MV('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in V_NEWTONRAPHSON_MV.M with the given
input arguments.
%
%      V_NEWTONRAPHSON_MV('Property','Value',...) creates a new
V_NEWTONRAPHSON_MV or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before v_newtonraphson_mv_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to v_newtonraphson_mv_OpeningFcn
via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
v_newtonraphson_mv

% Last Modified by GUIDE v2.5 24-Feb-2015 14:28:10

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @v_newtonraphson_mv_OpeningFcn,
...
                   'gui_OutputFcn',  @v_newtonraphson_mv_OutputFcn,
...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before v_newtonraphson_mv is made visible.
function v_newtonraphson_mv_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to v_newtonraphson_mv (see
VARARGIN)

% Choose default command line output for v_newtonraphson_mv
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes v_newtonraphson_mv wait for user response (see
UIRESUME)
% uiwait(handles.crear);


% --- Outputs from this function are returned to the command line.
function varargout = v_newtonraphson_mv_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in crear.
function crear_Callback(hObject, eventdata, handles)
% hObject    handle to crear (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
n=str2num(get(handles.n,'string'));
elem=cell(n,1);
elem(:,:)={''};
set(handles.tabla,'Data',elem);


% --- Executes on button press in solucion.
function solucion_Callback(hObject, eventdata, handles)
% hObject    handle to solucion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
A=cell(get(handles.tabla,'data'));
tol=str2double(get(handles.tol,'string'));
iter=str2double(get(handles.iter,'string'));
x0=str2num(get(handles.inicial,'string'));
```

```
M=sym(A);
[sol]=newtonsi(M,x0,tol,iter);
set(handles.sol,'string',num2str(sol));




function n_Callback(hObject, eventdata, handles)
% hObject    handle to n (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of n as text
%        str2double(get(hObject,'String')) returns contents of n as a
double


% --- Executes during object creation, after setting all properties.
function n_CreateFcn(hObject, eventdata, handles)
% hObject    handle to n (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function tol_Callback(hObject, eventdata, handles)
% hObject    handle to tol (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of tol as text
%        str2double(get(hObject,'String')) returns contents of tol as
a double


% --- Executes during object creation, after setting all properties.
function tol_CreateFcn(hObject, eventdata, handles)
% hObject    handle to tol (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function iter_Callback(hObject, eventdata, handles)
% hObject    handle to iter (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of iter as text
%        str2double(get(hObject,'String')) returns contents of iter as
a double


% --- Executes during object creation, after setting all properties.
function iter_CreateFcn(hObject, eventdata, handles)
% hObject    handle to iter (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function inicial_Callback(hObject, ~, handles)
% hObject    handle to inicial (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of inicial as text
%        str2double(get(hObject,'String')) returns contents of inicial
as a double


% --- Executes during object creation, after setting all properties.
function inicial_CreateFcn(hObject, eventdata, handles)
% hObject    handle to inicial (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function sol_Callback(hObject, eventdata, handles)
% hObject    handle to sol (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of sol as text
%        str2double(get(hObject,'String')) returns contents of sol as
a double


% --- Executes during object creation, after setting all properties.
function sol_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sol (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes when entered data in editable cell(s) in tabla.
function tabla_CellEditCallback(hObject, eventdata, handles)
% hObject    handle to tabla (see GCBO)
% eventdata  structure with the following fields (see UITABLE)
%   Indices: row and column indices of the cell(s) edited
%   PreviousData: previous data for the cell(s) edited
%   EditData: string(s) entered by the user
%   NewData: EditData or its converted form set on the Data property.
Empty if Data was not changed
%   Error: error string when failed to convert EditData to appropriate
value for Data
% handles    structure with handles and user data (see GUIDATA)
```